ABC INFORMATIQUE

N° 69

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Essayez l'Alice 90

Quatre tableurs analysés

Traduction des logiciels

Jeu : sur la Lune avec Atari

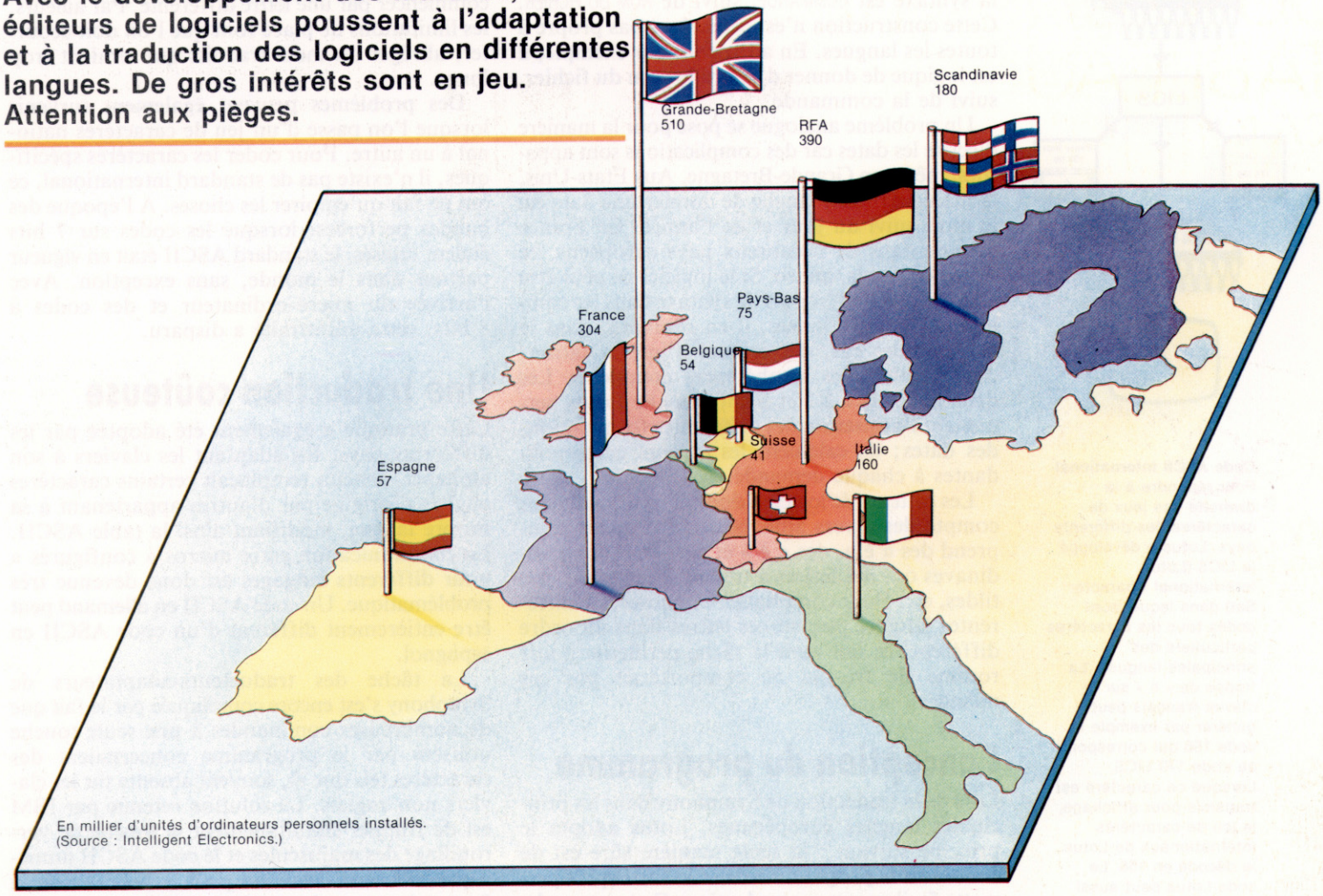
EDITIONS ATLAS

M 6062-69-12,50 F 90 FB - 3,80 FS - $ 1.95

Page manquante
(publicité et colophon)

He spricht anglais

Avec le développement des marchés, les éditeurs de logiciels poussent à l'adaptation et à la traduction des logiciels en différentes langues. De gros intérêts sont en jeu. Attention aux pièges.



En millier d'unités d'ordinateurs personnels installés.
(Source : Intelligent Electronics.)

L'anglais s'est immédiatement imposé comme langue de l'informatique. Pas étonnant : les Américains sont à l'origine, dans les années cinquante et soixante, du développement de cette « technique ».

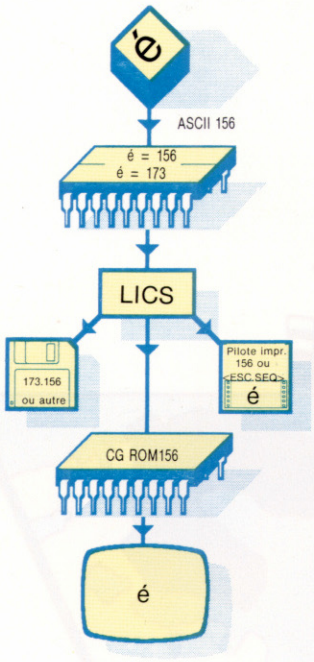
Si des langages informatiques furent créés par des équipes non anglophones — françaises en particulier — le développement des micro-ordinateurs venus d'outre-Atlantique a compliqué la situation des langages en Europe et au Japon ; les logiciels n'étaient pas traduits et adaptés. Les profits n'étaient pas suffisants pour investir dans des traductions. Le résultat de cette situation paradoxale fut que la Grande-Bretagne, qui avait l'avantage de partager la même langue que les États-Unis, devint le plus grand marché d'Europe pour les micro-ordinateurs.

Aujourd'hui, les éditeurs de logiciels européens, et en particulier français, ont relevé le défi. La concurrence s'est donc renforcée au point que les éditeurs anglais et américains n'hésitent plus à adapter leurs produits aux marchés étrangers qu'ils visent. Lotus Software, en tant que gros fournisseur de logiciels de gestion pour IBM, a été l'une des premières sociétés à mettre sur le marché des traductions de ses programmes.

Lotus 1-2-3 et Symphony figurent parmi les plus grosses ventes de « progiciels intégrés ». Ils comprennent généralement un tableur, une base de données, des fonctions de traitement de texte et des fonctions graphiques. Les données peuvent être transmises d'une application à l'autre, permettant par exemple de traiter avec le tableur des informations de la base de données. Le résultat étant ensuite intégré à un document.

La transcription-traduction d'un tel programme d'anglais en italien, par exemple, peut sembler facile. Les messages et commandes affichés à l'écran doivent être tous traduits. Pourtant, les difficultés commencent tout de suite. D'abord, le texte à traduire est intégré dans le code-source, son repérage dans les 120 K du code qui représente le texte et le programme exclusivement par des nombres n'est pas simple. Ensuite, si le texte traduit est plus long que le texte anglais, et c'est presque toujours le cas, le

PC de la CEE
Bénéficiant d'un énorme marché international en langue anglaise, les sociétés britanniques ont vite été les premières en Europe pour le nombre d'installations de micro-ordinateurs. Aujourd'hui, la tendance s'inverse. On estime généralement que la République fédérale allemande sera la première à dépasser la Grande-Bretagne en 1988 pour le parc de machines installées.
(Cl. Kevin Jones.)



Code ASCII international
Pour répondre à la diversité des jeux de caractères des différents pays, Lotus a développé le LICS (Lotus International Character Set) dans lequel sont codés tous les caractères particuliers des principales langues. La frappe de « é » sur le clavier français peut générer par exemple le code 156 qui correspond au code 173 LICS. Lorsque ce caractère est transmis pour affichage, le jeu de caractères internationaux de Lotus le décode en 156. Le code Lotus peut aussi parfois demander une séquence de commande pour un caractère particulier, tel que <è>, <ESC>, <ESPACE>, <>. Il est possible de sauvegarder du texte selon le code ASCII d'origine, en LICS ou selon les codes ASCII nationaux. (Cl. Ian McKinnell.)

programme demandera davantage d'octets. Les adresses du code venant après seront donc toutes changées, rendant inintelligibles les boucles et les appels de sous-programmes.

Autre problème : la syntaxe. Lorsqu'un utilisateur britannique veut intervenir sur un fichier, la syntaxe est COMMANDE, suivi de NOM DU FICHIER. Cette construction n'est cependant pas propre à toutes les langues. En allemand, par exemple, il est logique de donner d'abord le nom du fichier, suivi de la commande.

Un problème analogue se pose pour la manière d'écrire les dates car des complications sont apparues, même en Grande-Bretagne. Aux États-Unis, la manière traditionnelle de donner une date est le mois suivi du jour et de l'année. En France comme dans de nombreux pays européens, ce sera jour, mois, année. Si le logiciel ne peut être adapté à ces différences de syntaxe dans les commandes et les données, il en résultera, dans le meilleur des cas, des erreurs d'interprétation; dans le pire, des incohérences complètes. Les deux logiciels 1-2-3 et Symphony de Lotus permettent l'utilisation des différentes formulations des dates; ils calculent les valeurs correspondantes à chaque demande.

Les éditeurs de logiciels doivent également tenir compte des divers alphabets. Le français comprend des « é », des « è » et des « ê », les scandinaves ont des lettres à trémas, l'espagnol, des tildes, etc. Pour compliquer les choses, les différentes langues placent ces lettres dans un ordre différent, ce qui rend la tâche périlleuse à une routine de tri qui ne comporterait pas ces nuances.

Conception du programme

Lors de la traduction de Symphony dans les principales langues européennes, Lotus adopta le principe suivant : la seule manière sûre est de concevoir le programme de telle sorte qu'il permette facilement une traduction. Cette approche n'avait pas été celle du premier Lotus 1-2-3 et la société avait eu beaucoup de mal pour le traduire. Par la suite, Symphony a été traduit avec succès en français, en allemand et en langues scandinaves; de nouvelles versions en d'autres langues sont en préparation.

Afin de résoudre le problème de la localisation de la partie texte du code et de sa substitution par un nouveau texte de même longueur, Lotus adopta le principe d'une construction modulaire du programme. Ce dernier comprend alors deux parties : le code-source avec les routines du programme, et un segment de données contenant la zone texte.

Une telle localisation permet de résoudre deux des principales difficultés : disposer du texte dans un segment distinct de programme signifie que l'on peut ménager de la place en vue d'éventuelles différences de longueur entre les mots, et que le texte peut être retiré bien plus facilement du programme. Un programme utilitaire extrait les zones texte du code; elles peuvent alors être traduites et réinsérées. Enfin, avantage très appréciable, le texte peut être réécrit dans un autre ordre selon la syntaxe de la langue.

Mais il faut garder à l'esprit, lors de la traduction, que Symphony n'accepte comme commandes que les premières lettres des mots les désignant. Aussi, toutes les commandes doivent commencer par une lettre différente. Par ailleurs, les limitations de place font que l'on doit écourter, abréger les mots traduits qui seraient trop longs.

Des problèmes peuvent également survenir lorsque l'on passe d'un jeu de caractères national à un autre. Pour coder les caractères spécifiques, il n'existe pas de standard international, ce qui ne fait qu'empirer les choses. A l'époque des bandes perforées, lorsque les codes sur 7 bits étaient utilisés, le standard ASCII était en vigueur partout dans le monde, sans exception. Avec l'arrivée du micro-ordinateur et des codes à 8 bits, cette uniformité a disparu.

Une traduction coûteuse

Cette pratique a également été adoptée par les différents pays. En adaptant les claviers à son alphabet, chacun remplaçait certains caractères anglais d'origine par d'autres appartenant à sa propre langue, modifiant ainsi la table ASCII. La communication entre micros « configurés » pour différents langages est donc devenue très problématique. Un code ASCII en allemand peut être entièrement différent d'un code ASCII en espagnol.

La tâche des traducteurs-adaptateurs de Symphony s'est encore compliquée par le fait que de nombreuses commandes à une seule touche utilisées par le programme concernaient des caractères tels que @, souvent absents sur les claviers non anglais. La solution retenue par IBM est de frapper simultanément la touche de verrouillage des majuscules et le code ASCII numérique concerné. Mais on perd le bénéfice de la frappe à une seule touche!

Lotus a, pour sa part, décidé de contourner le problème en développant ses propres jeux de codes, appelés LICS (Lotus International Character Set). Ce jeu de deux cent cinquante caractères comprend toutes les lettres des principales langues européennes, et il figure sur chaque disquette Symphony. La traduction suppose de configurer le programme de sorte que les codes reçus depuis un clavier national soient transcrits en LICS. Pour simplifier l'affichage des caractères ne figurant pas aux claviers, Lotus utilise la frappe simultanée de la touche de verrouillage des majuscules et d'un chiffre compris entre 0 et 9.

La traduction d'un progiciel de gestion est une entreprise longue et coûteuse. Cela prend environ neuf mois et peut revenir à une somme allant de 100 000 à 1 000 000 de francs. Mais il est exclu d'ignorer le marché européen de quelques trois cents millions de clients potentiels. Les bénéfices que l'on peut attendre de la traduction des progiciels, tant pour les utilisateurs que pour les sociétés qui les développent, compensent largement l'effort d'investissement nécessaire.

Fantômes à vendre

Nous allons compléter le programme de notre jeu d'aventures avec un sous-programme chargé de « générer » au hasard des fantômes qui hanteront la forêt.

Un joueur abordant l'entrée d'un tunnel a le choix entre deux possibilités : y entrer ou reculer. S'il choisit la première solution, un nouveau sous-programme sera appelé en ligne 4655. Voyons maintenant le cas du sous-programme qui va permettre au joueur d'entrer dans le tunnel. Ce sous-programme est naturellement rédigé en fonction de certaines règles définies par le programmeur. Par exemple, on ne peut traverser le tunnel que si l'on dispose d'une lampe, et encore faut-il qu'elle soit allumée, sinon le joueur n'y verra rien.

Il doit, bien sûr, être capable d'entrer des instructions pendant qu'il se trouve à l'intérieur même du tunnel. C'est pourquoi, le sous-programme doit commencer par un module qui accueille toute commande et la décompose pour ensuite l'analyser.

Le joueur doit pouvoir se servir de verbes familiers — PRENDRE, LAISSER, LISTE ou FIN — mais, ici, il faudra se montrer prudent. Pour P (la variable qui gère sa position), le joueur est toujours à l'entrée du tunnel et peut donc ALLER dans les directions qui lui sont permises. Il faudra supprimer toutes les instructions ALLER tant qu'il sera dans le tunnel.

Si l'on repasse au sous-programme « commandes normales », une commande ALLER provoquera l'activation du drapeau « mouvement », MD, qui modifie la valeur de P. Le plus simple consiste donc à redonner à P la valeur qu'elle avait avant l'appel du sous-programme.

Le problème étant ainsi résolu, passons aux commandes particulières qui permettent de faire face à cette situation précise. La commande RECULER permettra au joueur de battre en retraite à l'entrée du tunnel. A l'intérieur de celui-ci, nous n'autoriserons que la commande LUMIÈRE, avec une variante, ALLUMER.

Si le joueur tape quoi que ce soit d'autre, il lui sera répondu chaque fois le même message assez explicite JE NE COMPRENDS PAS, puis le programme redemandera, par une boucle, une nouvelle instruction.

Les deux commandes LUMIÈRE et ALLUMER étant définies, il faudra procéder à diverses vérifications avant d'obéir à l'une d'elles :

1. L'objet spécifié est-il acceptable?
2. Le joueur l'a-t-il avec lui?
3. S'agit-il de la lampe?

Si la réponse à ces trois questions est « oui », le joueur pourra traverser le tunnel jusqu'à son autre sortie pour bien marquer que toutes les conditions relatives au passage ont bien été respectées. Vous remarquerez sans peine que nos procédures de vérification sont à peu près identiques à celles employées par les routines PRENDRE et LAISSER. Nous pourrons donc nous servir de sous-programmes déjà rédigés pour les mener à bien.

```
4700 REM ** ENTRER TUNNEL **
4705 SN$= "VOUS ENTREZ DANS LE TUNNEL MAIS IL FAIT TROP NOIR"
4710 SN$=SN$+ "POUR Y VOIR":GOSUB5500
4725 PRINT:INPUT"INSTRUCTIONS":IS$
4730 GOSUB2500:REM ANALYSE INSTRUCTION
4732 :
4735 IF F=0 THEN 4725:REM NON VALABLE
4740 OP=P:GOSUB3000:REM COMMANDES NORMALES
4745 IF MF=1THEN SN$= "IL FAIT SI NOIR QUE VOUS NE VOYEZ"
:P=OP
4747 IF MF=1THENSN$=SN$+ "QUE L'ENTREE DU TUNNEL":GOSUB5500
:MF=0:GOTO4725
4750 IF VF=1 THEN 4725:REM EXECUTION
4755 IF VB$="RECULER" AND P=4 THEN MF=1:P=6:RETURN
4760 IF VB$="RECULER" AND P=1 THEN MF=1:P=9:RETURN
4762 IFVB$<>"ALLUMER" ANDVB$<>"LUMIERE" THEN SN$="JE NE
COMPRENDS PAS
4765 IFVB$<>"ALLUMER" ANDVB$ >"LUMIERE" THEN GOSUB5500:GOT
O4725
4777 :
4780 REM ** S/P LAMPE **
4790 GOSUB5300:REM OBJET ACCEPTABLE ?
4795 OV=F:GOSUB5450:REM OBJET TRANSPORTE ?
4797 IF F=0 THEN SN$= "IL N'Y EN A PAS !"+W$:GOSUB5500:G
OTO4725
4800 IF HF=0 THEN SN$="VOUS NE L'AVEZ PAS !"
:GOSUB5500:GOTO4725
4810 REM ** EST-CE BIEN LA LAMPE ? **
4815 IF F<>2 THEN SN$="CA NE SERVIRA A RIEN !"
:GOSUB5500:GOTO4725
4835 REM ** CA MARCHE **
4840 SN$="VOUS ALLUMEZ LA LAMPE POUR GUIDER VOS PAS"
4845 SN$=SN$+"ET FINALEMENT RESSORTEZ A L'AUTRE BOUT DU
TUNNEL":GOSUB5500
4850 IF P=1 THEN MF=1:P=4:RETURN
4855 IF P=4 THEN MF=1:P=1:RETURN
```

Des événements plutôt surnaturels

Nous pouvons non seulement créer des lieux spéciaux, comme les entrées du tunnel, mais, de plus, programmer des périls aléatoires au sein de notre jeu d'aventures. Notre jeu s'appelle la Forêt hantée, mais nous n'y avons encore vu passer aucun fantôme, et ils n'ont jamais été signalés sur la carte. N'est-ce pas totalement paradoxal? En fait, le joueur les rencontrera au hasard, à mesure qu'il se déplace dans la forêt. Pour s'en débarrasser, il lui faudra agir de façon pour le moins inattendue.

Avant de regarder en détail le programme « fantômes », voyons d'abord comment générer de telles rencontres à l'intérieur du programme. La boucle principale de ce dernier appelle à la ligne 2700 un sous-programme pour tester si oui ou non un nouveau lieu doit se révéler spécial.

C'est aussi le meilleur endroit pour intégrer cette brève ligne qui décidera de l'apparition éventuelle d'un fantôme :

```
2707 REM ***** FANTOME ALEATOIRE *****
2710 IF P>4 AND RND(1)<0.1 THEN GOSUB 4290:RETURN
```

Le but de la ligne 2710 est de s'assurer que le lieu n'a rien de spécial; si c'était le cas, la présence d'un fantôme ne ferait que compliquer les choses inutilement. L'emploi de la commande RND donne alors une chance sur dix pour qu'il se manifeste.

RND génère des nombres « pseudo-aléatoires » — « pseudo » parce que l'ensemble des nombres générés est en fait prédéterminé. Pour que les choses soient un peu plus imprévisibles, nous utiliserons RND en conjonction avec un opérande négatif dans le cas du Commodore 64, ou la commande RANDOMISE du Spectrum (voir « Variantes de BASIC »).

```
207 R=RND(-1).
```

En cas d'appel de la routine « fantômes », nous passons à un autre scénario dans lequel le joueur doit faire face à l'apparition d'un fantôme. Le fonctionnement général de la routine est classique — affichage d'un message, demande d'instruction, analyse de la réponse.

Les commandes courantes sont gérées par le sous-programme habituel, mais, là encore, la commande ALLER est supprimée : le programme expliquera au joueur qu'il ne peut plus se déplacer car il est devenu paralysé par la terreur...

Filet de sécurité ou petit effort

Il est judicieux de doter la routine de commandes spéciales. C'est une question d'amour-propre : la qualité du jeu dépend de l'intérêt que vous porterez à développer le programme. Toutes ces commandes qui ne sont pas directement utiles se verront opposer un JE NE COMPRENDS PAS courtois mais ferme.

Cependant, il est plus intéressant encore de prévoir certaines réactions du joueur, même si elles se révèlent inopérantes.

Il pourrait, par exemple, décider de COMBATTRE ou de TUER le fantôme (mais peut-on tuer un esprit?). Dans les deux cas, il sera fait appel à un sous-programme qui se borne d'ailleurs à afficher un message précisant que cela ne sert à rien. C'est évidemment plus attractif que de répondre simplement JE NE COMPRENDS PAS ou quelque chose du même genre.

```
4290 REM **** S/P FANTOME ALEATOIRE ****
4295 SF=1:GC=0
4300 SN$="UN FRISSON GLACE VOUS PARCOURT"
4305 SN$=SN$+ "UN FANTOME TOUT BLANC"
4310 SN$=SN$+ "SORT DES TAILLIS ET"
4315 SN$=SN$+ "SE DIRIGE VERS VOUS":GOSUB5500:REM FO
RMAT
4325 SN$="IL SE RAPPROCHE":GOSUB5500
4330 GC=GC+1:IF GC>4 THEN GOSUB4455:REM
4335 PRINT:INPUT"INSTRUCTIONS";IS$
4340 GOSUB2500:REM ANALYSE INSTRUCTION
4345 IF F=0 THEN 4325:REM INSTRUCTION SUIVANTE
4350 OP=P:GOSUB3000:REM ANALYSE INSTRUCTION
4355 IF MF=1 AND VB$="ALLER"THEN GOSUB4400:GOTO 4325
4357 IF MF=1 AND VB$="REGARDER" THEN GOSUB2000:GOTO23
00:GOTO4325
4360 IF VF=1 THEN 4325:REM INSTRUCTION SUIVANTE
4365 REM ** NOUVELLES COMMANDES **
4370 IF VB$="TUER" OR VB$="COMBATTRE" THEN GOSUB4425:G
OTO 4325
4375 :
4385 IF VB$="CHANTER" THEN GOSUB4500:RETURN
4390 SN$="JE NE COMPRENDS PAS":GOSUB5500:GOTO4325
4395 :
4400 REM ** "ON NE BOUGE PAS ! **
4405 SN$='MORT DE FRAYEUR, VOUS NE POUVEZ"
4410 SN$=SN$+"BOUGER... ENFIN":MF=0:GOSUB5500:P=OP
4415 RETURN
4420 :
4425 REM ** COMBATTRE OU TUER **
4430 SN$="ETRE SURNATUREL, LE FANTOME"
4435 SN$=SN$+"SE RIT DE VOS VAINS EFFORTS"
4440 SN$=SN$+"POUR L'ATTEINDRE":GOSUB5500
4445 RETURN
4450 :
4455 REM ** ADIEU ! **
4460 SN$="UNE DOULEUR FULGURANTE"
4465 SN$=SN$+"VOUS COUCHE SUR LE SOL ET":GOSUB5500
4470 SN$="VOTRE AME QUITTE VOTRE CORPS INERTE"
4475 SN$=SN$+"POUR REJOINDRE"
4480 SN$=SN$+"LES ESPRITS TOURMENTES DE"
4485 SN$=SN$+"LA FORET HANTEE":GOSUB5500
4490 END
```

La touche finale

Si l'une quelconque des commandes normales ou des commandes sans effet sont entrées au clavier, le programme s'efforcera d'y obéir, puis redemandera une instruction. Il y a là une petite subtilité dans le déroulement de ce programme : il tient compte, en effet, du nombre d'instructions fournies par le joueur lorsqu'il est confronté au fantôme; s'il dépasse 4, le fantôme tuera l'intrus sans chercher plus loin. Le seul moyen pour le joueur de se sortir de là consiste à CHANTER une chanson.

Si cette option est effectivement choisie, trois chansons différentes lui sont proposées; l'une d'elles (choisie au hasard) pourra seule apaiser l'apparition du fantôme. En cas d'erreur, toutefois, l'âme du joueur ira rejoindre les esprits tourmentés qui errent dans la Forêt hantée...

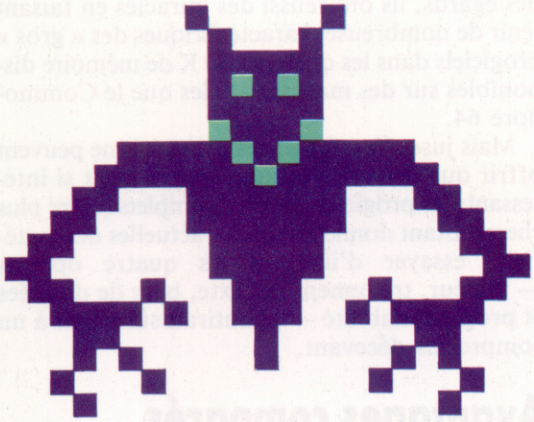
Comme quoi là aussi, tout finit par une chanson... mais la musique électronique n'adoucit pas toujours les mœurs.

```
4500 REM ** CHANTER **
4505 SN$="VOUS CONNAISSEZ TROIS CHANSONS. LAQUELLE
CHOISISSEZ-VOUS ?":GOSUB5500
4510 SN$="1>LE THEME DE "GHOSTBUSTERS'":GOSUB5500
4515 SN$="2>'RAMONA'":GOSUB5500
4520 SN$="3>'L'AUVERGNAT'":GOSUB5500
4525 PRINT:INPUT"FAITES UN CHOIX";C$
4530 IF VAL(C$)>3 OR VAL(C$)<1 THEN PRINT:PRINT"UN PEU DE
SERIEUX !"
4535 CR=INT(RND(1)*3+1
4537 IF CR<>VAL(C$) THEN GOSUB4542:REM PAS DE CHANCE !
4540 GOSUB4565:REM CORRECT
4542 REM **** S/P PAS DE CHANCE ****
4545 SN$="LE FANTOME DETESTE CET AIR ET"
4550 SN$=SN$+"S'APPROCHE DE VOUS":GOSUB5500
4555 GOSUB 4455:REM MORT
4560 :
4565 REM ** EXACT ! **
4570 SN$="CHARME PAR VOTRE CHANT, LE FANTOME"
4575 SN$=SN$+"DISPARAIT PEU A PEU ":GOSUB5500
4580 RETURN
```

Liz Dixon

Listage Digitaya

```
2690 IF P=37 THEN2780:REM TABLE VECTEURS
2700 IF P>7 THEN 2750:REM BOGUE ALEATOIRE

2740 REM ** BOGUE ALEATOIRE **
2750 RA=RND(TI)
2760 IF RA<0,05THEN GOSUB 5420:REM BOGUE
2770 RETURN
2780 REM ** TABLE VECTEUR **
2790 SF=1
2800 SN$ "VOUS FILEZ A VIVE ALLURE JUSQU'A UN NOUVEL
ENDROIT":GOSUB5880
2810 FORJ=1TO1000:NEXT:REM PAUSE
2820 P=INT(RND(TI)*40+7)
2830 MF=1:RETURN

4550 REM **** ULA ****
4560 SF=1
4570 RN=INT(RND(TI)*3+1)
4580 IF RN=1 THEN CD$="ET"
4590 IF RN=2 THEN CD$="OU"
4600 IF RN=3 THEN CD$="NON"
4610 SN$="ON VOIT SUR LE MUR 3 BOUTONS MARQUES"
4620 SN$=SN$+"'ET', 'OU' ET 'NON' ON PEUT ACCEDER A"
4630 SN$=SN$+"L'ACCUMULATEUR EN APPUYANT SUR LE BON"
4640 GOSUB5880:REM FORMAT
4650 :
4660 REM ** INSTRUCTIONS **
4670 PRINT:INPUT"INSTRUCTIONS";IS$
4680 GOSUB1700:GOSUB1900:REM ANALYSE
4690 IF MF=1THEN RETURN:REM SORTIR
4700 IF VF=1THEN 4670:REM INSTRUCTION SUIVANTE
4710 IFVB$="ALLUMER" OR VB$="APPUYER" THEN4740
4720 PRINT "JE NE COMPRENDS PAS":GOTO4670
4730 :
4740 REM ** COMMANDE VALABLE **
4750 IF VB$="APPUYER" THEN 4930
4760 REM ** COMMANDE 'ALLUMER' **
4770 GOSUB5730:REM OBJET ACCEPTABLE ?
4780 IFF=0THENPRINT "IL N'Y EN A PAS !";NN$:GOTO4670:REM
 NEXT INSTRUCTION
4790 :
4800 REM ** OBJET : LIVRE DU CODE ? **
4810 IF F=7 THEN4850:REM OK
4820 SN$="CELA NE SERT A RIEN":GOSUB5880
4830 GOTO4670:REM "INSTRUCTION SUIVANTE"
4840 :
4850 OV=7:GOSUB5830:REM AVEZ VOUS L'OBJET ?
4860 IFHF=1THEN4900:REM OK
4870 SN$="VOUS NE L'AVEZ PAS !"
4880 GOSUB5880:GOTO4670:REM INSTRUCTION SUIVANTE
4890 :
4900 SN$="VOUS OUVREZ LE LIVRE DU CODE ET TROUVEZ LE MOT'"
+CD$+"'ECRIT A L'INTERIEUR"
4910 GOSUB5800:GOTO4670:REM INSTRUCTION SUIVANTE
4920 :
4930 REM ** COMMAND IS PRESS **
4940 IFNN$="ET"OR NN$="OU" OR NN$="NON" THEN4970
4950 SN$="IL N'Y EN A PAS !"+NN$:GOSUB5880:GOTO4670:REM
INSTRUCTION SUIVANTE
4960 :
4970 REM ** VRAI OU FAUX **
4980 IFNN$=CD$ THEN GOSUB5100:RETURN
4990 GOSUB5010:RETURN
5000 :
5010 REM ** S/P FAUX **
5020 SN$="FAUX UNE TRAPPE S'OUVRE ET VOUS VOILA"
5030 SN$=SN$+"RENVOYE EN MEMOIRE"
5040 GOSUB5880:REM FORMAT
5050 IF RN=1 THEN P=39
5060 IF RN=2 THEN P=35
5070 IF RN=3 THEN P=29
5080 MF=1:RETURN
5090 :
5100 REM ** S/P VRAI **
5110 SN$="LA PORTE DE L'ACCUMULATEUR S'OUVRE ET"
5120 SN$=SN$+"VOUS LA FRANCHISSEZ":GOSUB5880
5130 P=30:MF=1:RETURN

5420 REM **** BOGUE ALEATOIRE ****
5430 SF=1
5440 SN$="UNE HORRIBLE BOGUE SURGIT DE DERRIERE UNE PUCE"
5450 SN$=SN$+"ET RAMPE VERS VOUS":GOSUB5880
5460 :
5470 REM ** INSTRUCTIONS **
5480 PRINT:INPUT"INSTRUCTIONS";IS$
5490 GOSUB1700:GOSUB1900:REM ANALYSE
5500 IFMF=1THENMF=0:PRINT"VOUS NE POUVEZ PAS BOUGER":GOTO5480
5510 IF VF=1THEN5480:REM INSTRUCTION SUIVANTE
5520 IF VB$="TUER"ORVB$="COMBATTRE"THEN5550
5530 PRINT"JE NE COMPRENDS PAS":GOTO5480
5540 :
5550 REM ** TUER OU COMBATTRE **
5560 RA=RND(TI)
5570 IFRA<0.5 THEN GOSUB5600
5580 GOSUB5670:RETURN
5590 :
5600 REM **** PERDU ! ****
5610 SN$="VOUS VOUS BATTEZ AVEC LA BOGUE QUI"
5620 SN$=SN$+"VOUS ACCABLE D'ERREURS DE PROGRAMMATION"
5630 SN$=SN$+"ET VOTRE TETE EXPLOSE"
5640 GOSUB5880
5650 END
5660 :
5670 REM **** GAGNE ! ****
5680 SN$="VOUS COMBATTEZ LA BOGUE ET APRES BIEN DES EFFORTS"
5690 SN$=SN$+"VOUS LA TERRASSEZ":GOSUB5880
5700 RETURN
```



Variantes de basic

Spectrum :

Dans les deux programmes, remplacez SN$ par S$, IS$ par T$, IV$(,) par V$(,), VB$ par B$, CD$ par C$ et NN$ par R$.
Glissez ces lignes dans la Forêt hantée :

```
207 RAND
4815 IF F<>2 THEN LET S$="LE ":LET
  A$=V$(F,1):GOSUB7000
4816 IF F<>2 THEN LET S$=S$+"NE SERT
  A RIEN":GOSUB5500:GOTO4725
```

Et dans Digitaya :

```
2750 LET RN = RND(1)
2820 LET P = INT(RND(1)*40+7)
4570 LET RN = INT(RND(1)*3+1)
4820 LET SN$="VOTRE":LET A$ = V$(F,1):GOSUB8500
4825 LET SN$ = SN$+"NE SERT A RIEN":GOSUB5880
5560 LET RA = RND(1)
```

Les puissants

Nous allons examiner quatre programmes basés sur des tableurs : Micro Swift, Practicalc II, PS et Vizastar. Ces produits confirment que les micros personnels ont de quoi tenir tête aux micros de gestion.

Micro Swift, Practicalc II, PS et Vizastar appartiennent à une nouvelle gamme de progiciels avancés fondés sur des tableurs; ils se sont manifestement inspirés de Lotus 1-2-3 et de son successeur, de Symphony. Mais une différence notable existe : ces derniers, écrits pour l'IBM PC et ses compatibles, supposent 296 K de mémoire utilisateur pour Lotus et au moins 320 K pour Symphony. Les nouveaux progiciels sont écrits, quant à eux, pour micros domestiques, et, à bien des égards, ils ont réussi des miracles en faisant tenir de nombreuses caractéristiques des « gros » progiciels dans les quelques 30 K de mémoire disponibles sur des machines telles que le Commodore 64.

Mais jusqu'à présent, ces progiciels ne peuvent offrir que deux des options qui rendent si intéressants les progiciels les plus complets (et les plus chers). Étant donné les limites actuelles des matériels, essayer d'intégrer les quatre options — tableur, traitement de texte, base de données et programmabilité — aboutirait sûrement à un compromis décevant.

Avantages comparés

Abordons, pour apprécier les avantages respectifs de ces quatre programmes, quelques-unes de leurs options. PS, Micro Swift et Vizastar sont tous programmables, à des degrés divers. C'est là une propriété précieuse, puisqu'elle permet de rendre automatiques des fonctions qui, autrement, supposeraient de nombreuses frappes. La démarche des trois programmes pour créer ces fonctions automatiques est différente, et nous allons maintenant distinguer chacun des cas.

Les modules sont programmés, avec PS, par le biais de commandes BASIC normales. Les modules sont ensuite sauvegardés en faisant <f3>, et exécutés par <U>. Ils peuvent aussi s'auto-exécuter au chargement du programme, si on les sauvegarde (SAVE) sur disque, en mentionnant un point après le nom du programme. PS dispose de toute une gamme de caractéristiques très utiles : il peut, par exemple, susciter un branchement sur un sous-programme (GOSUB), depuis une formule occupant une position du tableur. Les fonctions se définissent par la fonction FN, le programme pouvant également transmettre des chaînes de caractères, des colonnes et des lignes, ainsi que des valeurs numériques.

Pour programmer Micro Swift, il suffit de mettre une liste de commandes dans la colonne Z. La première commande donne le nom du programme et est précédée du signe dièse (#). La dernière ligne porte la commande @QUIT. Prenons un exemple simple :

```
Z1 #SUM
Z2 @SUM(A1,A3)
Z3 @ASSIGN(Z2,A4)
Z4 @QUIT
```

Ce programme additionne les valeurs présentes aux positions A1, A2 et A3, et assigne la valeur obtenue en Z2, à A4. Le programme est appelé à l'aide de l'instruction #SUM.

De tous les progiciels abordés ici, le plus simple à programmer est peut-être Vizastar. En effet, les commandes sont constituées des premières lettres que l'on aurait tapées pour les exécuter manuellement. Ainsi, pour utiliser une base de données déterminée, vous frapperiez la touche CBM suivie de D(onnées), U(tiliser), D(onnées), et enfin, le nom de la base de données, et retour-chariot. En programmation, le signe / est utilisé à la place de la touche CBM, de sorte que / DUDnom[RC] exécute l'action voulue lorsque l'on appuie sur <f8>. Les touches de fonction et d'édition se programment en appuyant sur <CTRL> et la touche appropriée. La lettre en question figure dans les programmes utilisant la fonction correspondante. Lorsque les touches de déplacement du curseur sont programmées de la sorte, elles s'affichent [haut], [bas], [gauche] et [droite].

La base de données de Vizastar est puissante; elle utilise une partie du tableau virtuel (à partir des lignes 1000, et non accessible à l'utilisateur), pour sauvegarder les formats d'enregistrement. Chacun peut comprendre jusqu'à neuf écrans accessibles par les commandes Touche, Suivant, Précédent, Dernier ou Courant (chacune utilisant la lettre initiale d'une option d'un menu de commandes). Les enregistrements peuvent aussi être Ajoutés, Modifiés ou Détruits.

Les zones portent des noms de lettres, de A jusqu'à BK, ces noms se rapportant aux colonnes de mêmes noms dans le tableau. Ainsi, les critères de recherche peuvent figurer sur une ligne blanche du tableau. A est toujours la zone clé de recherche, la zone sur laquelle les données sont triées.

Practicalc II est un tableur qui permet au texte de déborder d'une position sur sa voisine vide, caractéristique appelée « long identificateur ». Le programme fonctionne ainsi comme un traitement de texte avec longueur maximale de ligne de cent caractères. La plupart des propriétés d'un traitement de texte sont alors réunies, telles que

Chargements poids lourds

La plupart des entreprises de transport disposent d'environ cinq véhicules. Les progiciels disponibles pour gérer ces effectifs ne s'appliquent qu'au-delà de cent véhicules et coûtent environ 10 000 F. Le progiciel de gestion informatisée d'effectifs MEM, par exemple, coûte environ 15 000 F, plus 1 000 F pour l'équipement de chaque véhicule en modules d'analyse des coûts et des vitesses.

On comprend pourquoi peu d'entreprises de transport sont encore informatisées. Cela dit, après enquête, un système correspondant davantage à la réalité du marché des camionneurs a été développé. Tout en profitant des possibilités offertes par des progiciels de grande capacité, un projet vit le jour avec l'un des micros les plus répandus du marché, le Commodore 64, en utilisant Vizastar, tableur programmable avec base de données, qui coûte moins de 1 000 F. Le coût total du système revenait à 10 000 F environ, logiciel et matériel compris, soit le cinquième du coût total des systèmes plus grands.

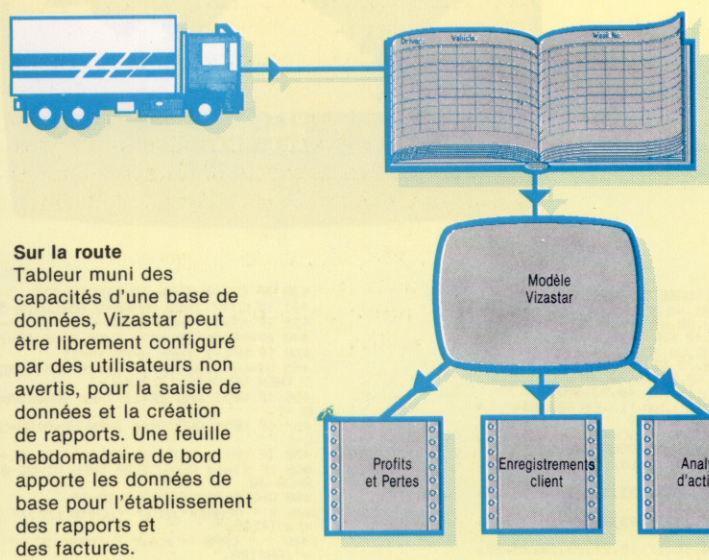
En Grande-Bretagne, une enquête fut entreprise auprès des petits camionneurs pour savoir si un tel système informatique était utile, et s'ils étaient prêts à investir. Des formulaires, bien connus des transporteurs, furent alors remplis (les missions accomplies, les voyages, les destinations, les kilométrages, les coûts en carburant, les frais et les coûts de fonctionnement). L'opération fut un succès.

Depuis, à la fin de chaque semaine, les données en provenance de ces feuilles de route sont saisies au tableur.

Dès la fin de la saisie des données, le tableur calcule si ce dernier exercice se solde par un profit ou une perte. Le logiciel produit alors une analyse complète de l'état des affaires de la semaine. Ce dernier peut être consolidé en analyses mensuelles, puis en états annuels.

Comme Vizastar utilise une partie du tableur en tant que base de données, la sauvegarde et la transmission d'enregistrements sur et à partir de disques reviennent exactement au même qu'avec les programmes de gestion de bases de données spécialisées (utilisation d'une zone-clé, recherche dans la liste par les commandes Suivant, Antérieur ou Courant).

Il est possible de gérer un enregistrement client permanent. Le tableau peut également servir en référence.



Sur la route
Tableur muni des capacités d'une base de données, Vizastar peut être librement configuré par des utilisateurs non avertis, pour la saisie de données et la création de rapports. Une feuille hebdomadaire de bord apporte les données de base pour l'établissement des rapports et des factures.
(Cl. Ian McKinnell.)

l'enroulement horizontal de la frappe, le déplacement de blocs, l'insertion et la suppression.

Il est également possible de charger (LOAD) un tableau dans une partie d'un document. Le tableau reste alors « actif », ce qui signifie que ses formules, valeurs et autres contenus peuvent être modifiés dans le cadre du document principal, sans affecter, bien sûr, le tableau sur disque.

Bien que les restrictions de mémoire limitent à quelques options vitales ces quatre programmes, ils peuvent néanmoins avoir accès à des fichiers de traitement de texte ou de base de données en provenance de progiciels conçus par leurs sociétés respectives. Vizastar peut gérer les fichiers de traitement de texte obtenus à partir de Vizawrite; Micro Swift permet l'accès aux fichiers base de données de Micro Magpie; Practical et PS peuvent utiliser les fichiers de Practifile. En outre, comme leurs fichiers utilisent tous un mode d'accès séquentiel, les quatre programmes peuvent échanger entre eux des fichiers, mais aussi avoir accès à des programmes sans aucun rapport, tels que le traitement de texte Easy Script. S'il ne s'agit pas tout à fait d'intégration logicielle, cela n'en est pas loin.

Micro Swift : pour le Commodore 64.
Prix : 200 F environ.
Éditeur : Audiogenic.
Format : disquette.

Practicalc II : pour l'Apple II 48 K et le Commodore 64.
Prix : 800 F environ.
Éditeur : Practicorp.
Format : disquette.

PS : pour le Commodore 64.
Prix : 800 F environ.
Éditeur : Practicorp.
Format : disquette.

Vizastar : pour le Commodore 64.
Prix : 1 200 F environ.
Éditeur : Viza Software.
Format : disque à cartouche 4 K.

Sur la lune

Ce jeu a été écrit par Paul Dunning pour le micro-ordinateur Atari. Le manche à balai est indispensable pour poser votre vaisseau spatial sur la lune.

Il vous faut poser votre vaisseau spatial en toute sécurité sur l'une des trois aires bleues. Si votre vitesse est trop élevée, vous vous écrasez; une vitesse de 5 semble bonne. Votre score est calculé en fonction de la vitesse à laquelle vous atterrissez et de l'aire d'atterrissage que vous choisissez. Après un atterrissage réussi, vous repartez avec un bonus de cinq cents litres de carburant. Le jeu se termine lorsque vous avez épuisé votre stock de carburant.

```
0 REM * ATTERRISSAGE DE PAUL DUNNING *
1 DATA 108,60,81,40,53,40,60,40,64,40,72
,40,40,40,53,40,60,40,64,40,72,40,40,40,
53,40,60,40,64,40,60,40,72,60
2 DATA 108,60,96,80,60,40,64,40,72,40,81
,80,72,40,64,40,72,60,96,40,85,40,53,80,
40,40,45,40,50,40,53,40,60,40
3 DATA 68,40,72,40,81,40,53,80,-1
4 CLR :DIM A$(100),H$(11),E$(60):H$="  a
tari":POKE 752,1:RE=500
5 AA=1:CA=4000:TE=100:SC=0:AN=0:VI=AN:PO
KE 53278,1:GOSUB 900:GOSUB 4000
6 DATA 11,12,13,13,14,15,17,19,21,22,23,
23,24
7 DATA 25,26,29,30,31,33,33,33,33,33,33,
31
8 DATA 31,30,29,28,26,25,25,24,23,21,18,
16,15
9 DATA 14,14,14,15,17,19,21,23,24,25,26,
27,29
10 DATA 29,29,29,29,27,26,26,25,26,27,28
,28
11 DATA 27,26,25,25,23,23,23,23,23,23,23
,21,21
12 DATA 20,18,17,17
29 FOR I=1 TO 8:POKE 53247+I,0:NEXT I:GO
TO 495
30 END
120 REM POKE 53248,30
200 REM *****  JEU JOUEUR   *****
210 POKE 559,0:POKE 710,148:POKE 712,0
215 POKE 752,1:AL=110
230 RESTORE 235:FOR S2=1 TO 60:READ S3:E
$(S2,S2)=CHR$(S3):NEXT S2
235 DATA 104,104,104,74,72,176,11,160,2,
177,203,136,145,203,200,200,208,247,104,
74,72
236 DATA 176,11,160,253,177,203,200,145,
203,136
237 DATA 136,208,247,104,74,72,176,7,198
,205
238 DATA 165,205,141,0,208,104,74,176,7,
230,205,165,205,141,0,208,96,0,0
240 A=PEEK(106)-16:POKE 54279,A:POKE 204
,A+4:POKE 203,0:PMBASE=A*256
250 POKE 53277,3:POKE 704,40:POKE 53248,
150:POKE 205,RND(1)*130+70:POKE 53256,0
260 FOR I=PMBASE+1024 TO PMBASE+1280:POK
E I,0:NEXT I
270 RESTORE 290
280 FOR I=PMBASE+1155 TO PMBASE+1209:REA
D B:POKE I,B:NEXT I
290 DATA 8,8,60,60,126,126,195,195,126,1
26,60,60,24,24,126,126,165,165
300 DATA 165,165,165,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0
310 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0
320 POKE 559,62:GOTO 390
370 ST=STICK(0)
380 CO=PEEK(53252):IF CO>0 THEN GOSUB 50
0
385 POKE 53278,1
390 IF AA=1 THEN FOR Q=0 TO 100:A=USR(AD
R(E$),14):SOUND 0,255,8,15:NEXT Q:AA=0:A
=USR(ADR(E$),11):SOUND 0,0,0,0
400 POKE 53278,3
410 IF ST=14 AND CA>=30 THEN SOUND 0,255
,8,15:GOTO 420
415 SOUND 0,0,0,0
420 IF ST=7 OR ST=11 AND CA>=5 THEN SOUN
D 1,100,8,3:GOTO 430
425 SOUND 1,0,0,0
430 IF CA<=0 THEN CA=0:GOTO 455
431 IF ST=7 THEN AN=AN+0.05:CA=CA-5
432 IF ST=11 THEN AN=AN-0.05:CA=CA-5
440 HP=HP+AN:IF HP>1 THEN HP=0:SR=7
450 IF HP<-1 THEN HP=0:SR=11
455 VI=VI+0.02:VP=VP+VI:IF VP>1 AND SR=1
5 THEN VP=0:SR=13:AL=AL-1
456 IF VP>1 AND SR=11 THEN VP=0:SR=9:AL=
AL-1
457 IF VP>1 AND SR=7 THEN VP=0:SR=5:AL=A
L-1
458 IF VP<-1 THEN VP=0:SR=14:AL=AL+1
460 IF ST=14 AND CA>29 THEN VI=VI-0.05:C
A=CA-30
480 A=USR(ADR(E$),SR):SR=15
485 ? " SCORE=";SC;" ";CHR$(127);"  VITE
=";VI*100;"  "
486 ? " CARB =";CA;" ";CHR$(127);"  ANGL
=";AN*100;"  "
487 ? "TEMPS =";INT(TE);" ";CHR$(127);"
 ALT =";AL;" "
488 ? CHR$(28);CHR$(28);CHR$(28);
489 TE=TE-0.1
490 IF TE=0 THEN 720
492 GOTO 370
495 GOTO 210
500 IF CO=4 AND VI<0.5 THEN 600
505 IF CO=4 AND VI>0.4 THEN 700
510 IF CO=2 THEN 700
520 RETURN
600 FOR Q=1 TO 2:FOR W=255 TO 100 STEP -
1
610 SOUND 0,W,10,15:SOUND 1,W-10,10,15:S
OUND 2,W-20,10,15
620 NEXT W:NEXT Q
630 SS=0.5-VI:SO=SC:IF SS=0 THEN SC=SC+5
0:GOTO 650
640 IF AL<5 THEN SC=SC+(SS*100)*10:GOTO
650
642 IF AL<20 THEN SC=SC+(SS*100)*20:GOTO
 650
644 IF AL<60 THEN SC=SC+(SS*100)*5
650 IF CA<2500 THEN CA=CA+(SC-SO*2)
655 FOR Q=0 TO 3:SOUND Q,0,0,0:NEXT Q:AA
=1:POKE 53278,1:GOSUB 800:GOTO 210
700 FOR Q=0 TO 50:POKE 712,RND(1)*255:SO
UND 0,255,8,15:SOUND 1,240,8,15:SOUND 2,
220,8,15:NEXT Q
705 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,
0,0,0
706 IF CA<30 THEN CA=0:GOTO 720
710 POKE 712,0:AA=1:GOSUB 800:GOTO 210
720 POKE 712,0:GOTO 810
800 VP=0:HP=0:AN=0:VI=0:TE=100:RETURN
810 IF SC>RE THEN GOSUB 1010
820 GRAPHICS 17:SETCOLOR 4,1,0
830 ? #6;"     TERMINE"
835 ? #6:? #6;"       RECORD ";RE
836 ? #6:? #6;"     ";H$
840 ? #6:? #6;"     TAPEZ START "
850 POKE 53248,0
860 IF PEEK(53279)=6 THEN RESTORE :GOTO
5
870 GOTO 860
899 END
900 REM INTRO
910 GRAPHICS 17:SETCOLOR 4,1,0:E=0
920 ? #6;"     ATTERRISSAGE "
930 ? #6:? #6;" DE P.DUNNING SEP.82"
940 ? #6:? #6;" MANCHE A BALAIS (0)"
950 ? #6:? #6;"          HAUT     "
960 ? #6:? #6;"              ^"
970 ? #6;" GAUCHE<     >DROITE"
980 ? #6
985 POKE 53248,0
990 READ M:IF M<0 THEN 998
992 READ P
995 SOUND 0,M,10,15:FOR PP=0 TO P:NEXT P
P
997 GOTO 990
998 ? #6;"    bonne chance"
999 FOR X=1 TO 13:FOR Y=0 TO 255 STEP 20
:SOUND 0,Y,10,15:NEXT Y:NEXT X:SOUND 0,0
,0,0:RETURN
1000 RESTORE 1:GOTO 990
1010 H$=" ":TE=30:GRAPHICS 18:SETCOLOR 4
,1,0:? #6;"          PAS MAL ":? #6:X=1:Y=
1:RE=SC
1011 ? #6;"  a b c d e f g h "
1012 ? #6
1013 ? #6;"  i j k l m n o p "
1014 ? #6
1015 ? #6;"  q r s t u v w x "
1016 ? #6
1017 ? #6;"  y z . -     R E "
1018 POKE 53248,0
1020 FOR Q=10 TO 245:SOUND 0,Q-10,8,15:S
OUND 1,Q,8,15:SOUND 2,Q+10,8,15:NEXT Q
1030 POSITION X,Y:? #6;"XXX":POSITION X,
Y+1:? #6;"X":POSITION X+2,Y+1:? #6;"X":P
OSITION X,Y+2:? #6;"XXX"
1040 IF STICK(0)<>7 THEN 1050
1045 POSITION X,Y:? #6;"   ":POSITION X,
Y+1:? #6;" ":POSITION X+2,Y+1:? #6;" ":P
OSITION X,Y+2:? #6;"   "
1046 IF X>13 AND Y=7 THEN X=1:Y=1:GOTO 1
050
1047 IF X>13 THEN X=1:Y=Y+2:GOTO 1050
1048 X=X+2
1050 IF STICK(0)<>11 THEN 1060
1055 POSITION X,Y:? #6;"   ":POSITION X,
Y+1:? #6;" ":POSITION X+2,Y+1:? #6;" ":P
OSITION X,Y+2:? #6;"   "
1056 IF X<3 AND Y=1 THEN X=15:Y=7:GOTO 1
060
1057 IF X<3 THEN X=15:Y=Y-2:GOTO 1060
1058 X=X-2
1060 POSITION 6,10:? #6;"TEMP ";INT(TE);
" ":TE=TE-0.1:IF TE=0 THEN 1100
1065 FOR PP=0 TO 20:NEXT PP
1069 IF STRIG(0)<>0 THEN 1080
1071 LOCATE X+1,Y+1,XX:IF XX=ASC("E") TH
EN 1100
1072 IF XX=ASC("R") AND LEN(H$)>1 THEN H
$=H$(1,LEN(H$)-1):GOTO 1080
1073 IF LEN(H$)>10 THEN 1080
1075 H$(LEN(H$)+1)=CHR$(XX)
1080 POSITION 4,11:? #6;H$;" "
1099 GOTO 1030
1100 FOR Q=0 TO 3:SOUND Q,0,0,0:NEXT Q
1110 RETURN
4000 GRAPHICS 5
4010 RESTORE 6:COLOR 2
4020 FOR X=0 TO 79
4030 READ Y
4040 PLOT X,39:DRAWTO X,Y
4050 NEXT X
4060 COLOR 3
4070 PLOT 18,32:DRAWTO 23,32
4080 PLOT 50,28:DRAWTO 54,28
4090 PLOT 67,22:DRAWTO 73,22
4100 RETURN
```

© SYBEX 1984

Alice a grandi

L'Alice 90, le nouvel ordinateur familial de Matra, a su conserver la simplicité d'accès à la micro-informatique d'Alice, tout en disposant de possibilités beaucoup plus larges.

Matra reste fidèle à la couleur rouge pour la famille de ses ordinateurs domestiques Alice. Avec ses 32 K de mémoire utilisateur, on est loin des « petites » possibilités du premier Alice 4 K. (Cl. Matra.)

Un an après avoir lancé la première version d'Alice 4 K, Matra met aujourd'hui sur le marché un nouvel ordinateur domestique : Alice 90. Comme son prédécesseur et la variante Alice 32, le nouvel Alice est produit à Colmar.

De la même couleur rouge que les autres membres de sa famille, Alice 90 représente, par rapport aux précédents modèles, une considérable amélioration de qualité et de fiabilité due à un niveau d'intégration plus poussé de ses composants. En effet, tout en offrant huit fois plus de capacité mémoire, Alice 90 utilise moins de composants que l'Alice 4 K. Il dispose aussi de possibilités beaucoup plus larges.

La mémoire morte, d'une capacité de 16 K, comprend un BASIC Microsoft cent pour cent compatible avec l'Alice 4 K. Les 32 K de mémoire utilisateur offrent de très larges possibilités de programmation.

Ces nouvelles qualités font de l'Alice 90 une machine proche du matériel professionnel pour un prix très compétitif.

A sa ligne pure et sobre s'ajoutent des qualités ergonomiques, non négligeables même si l'ordinateur n'est pas utilisé dans un contexte professionnel. Son clavier incliné est spécialement étudié pour un meilleur confort de frappe.

L'électronique de l'incrustation vidéo est développée dans cette machine. Elle permettra de suivre les émissions d'initiation de la télévision française grâce à une extension logicielle.

Un familial haut de gamme

Alice 90 est, comme les autres ordinateurs de la famille Matra, construit autour du microprocesseur 6803 de Motorola. Sa mémoire, de 56 K au total, se décompose en 40 K de mémoire vive (RAM), dont 32 sont disponibles pour l'utilisateur et 8 K pour l'écran, ainsi que 16 K de mémoire morte (ROM) pour le BASIC Microsoft résident, comprenant un éditeur et un éditeur-assembleur.

Alice 90 offre différentes possibilités d'affichage sur écran : le texte peut être affiché soit sur 16 lignes et 32 colonnes, soit sur 25 lignes, avec l'alternative de 40 ou 80 colonnes. L'écran graphique comporte la définition moyenne de 160 × 125 pixels, qui peut être affinée à l'aide de l'assembleur pour atteindre 320 × 250 pixels.

Comme tout ordinateur familial digne de ce nom, Alice 90 se connecte sur tout téléviseur couleur muni de la prise Péritel. En revanche, il ne dispose pas de l'adaptateur noir et blanc qui permettait de connecter l'Alice à un téléviseur dépourvu de cette prise.

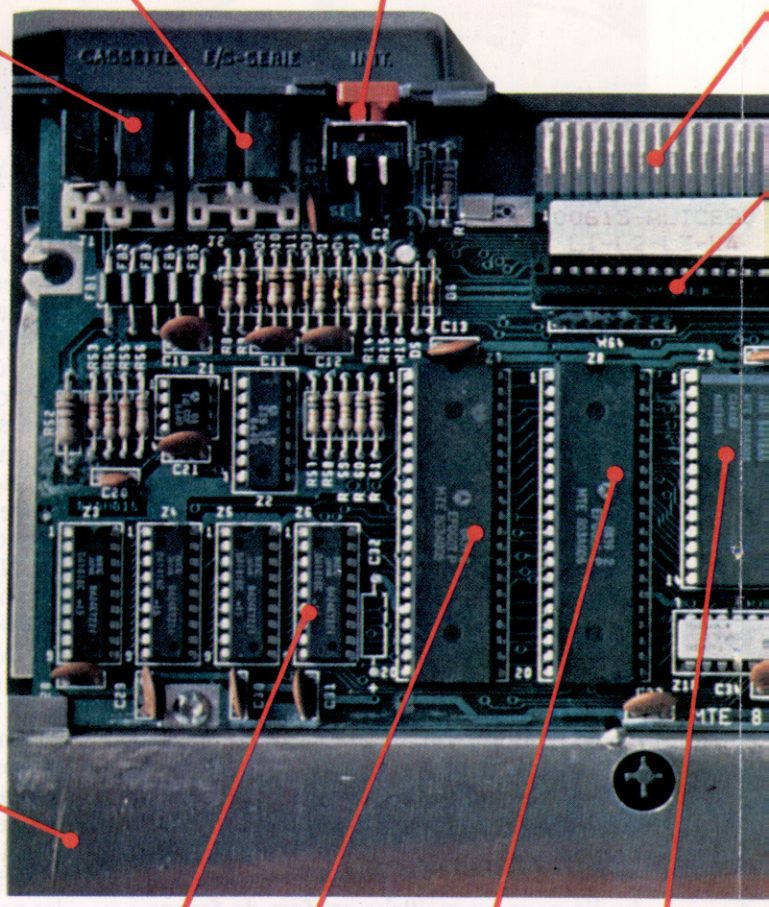
Alice 90 dispose d'une sortie imprimante qui autorise l'utilisation de l'imprimante conçue pour les premières versions d'Alice. Ainsi, les « mordus » de la programmation pourront lister leurs programmes dans le but de les améliorer. Il s'agit d'une imprimante thermique d'une couleur rouge vif assortie à celle d'Alice 90. Sa taille et son

A partir de ce clavier, vous pouvez faire de l'incrustation vidéo, c'est-à-dire suivre les émissions d'initiation à l'informatique de la télévision tout en affichant vos propres informations. (Cl. Matra.)

design sont aussi très adaptés au micro-ordinateur. D'un coût abordable, l'imprimante Alice devient très vite le périphérique indispensable au système Alice.

Ses caractéristiques sont assez modestes, mais en rapport avec son coût modique : la vitesse d'impression est de 30 caractères par seconde, sur 32 colonnes de caractères normaux, double largeur ou semi-graphiques. La vitesse de transfert est de 600 bits par seconde. L'imprimante Alice est livrée avec son câble de raccordement sur l'interface RS232 d'Alice 90.

Encore plus nécessaire que l'imprimante, le lecteur-enregistreur de Matra-Hachette complète la gamme des périphériques pour Alice. Aussi rouge que les autres produits de la gamme, et livré avec son câble, il assure une grande fiabilité et une qualité de transfert des données. Il comprend un arrêt automatique fin de bande. Des diodes électroluminescentes indiquent l'état de marche.

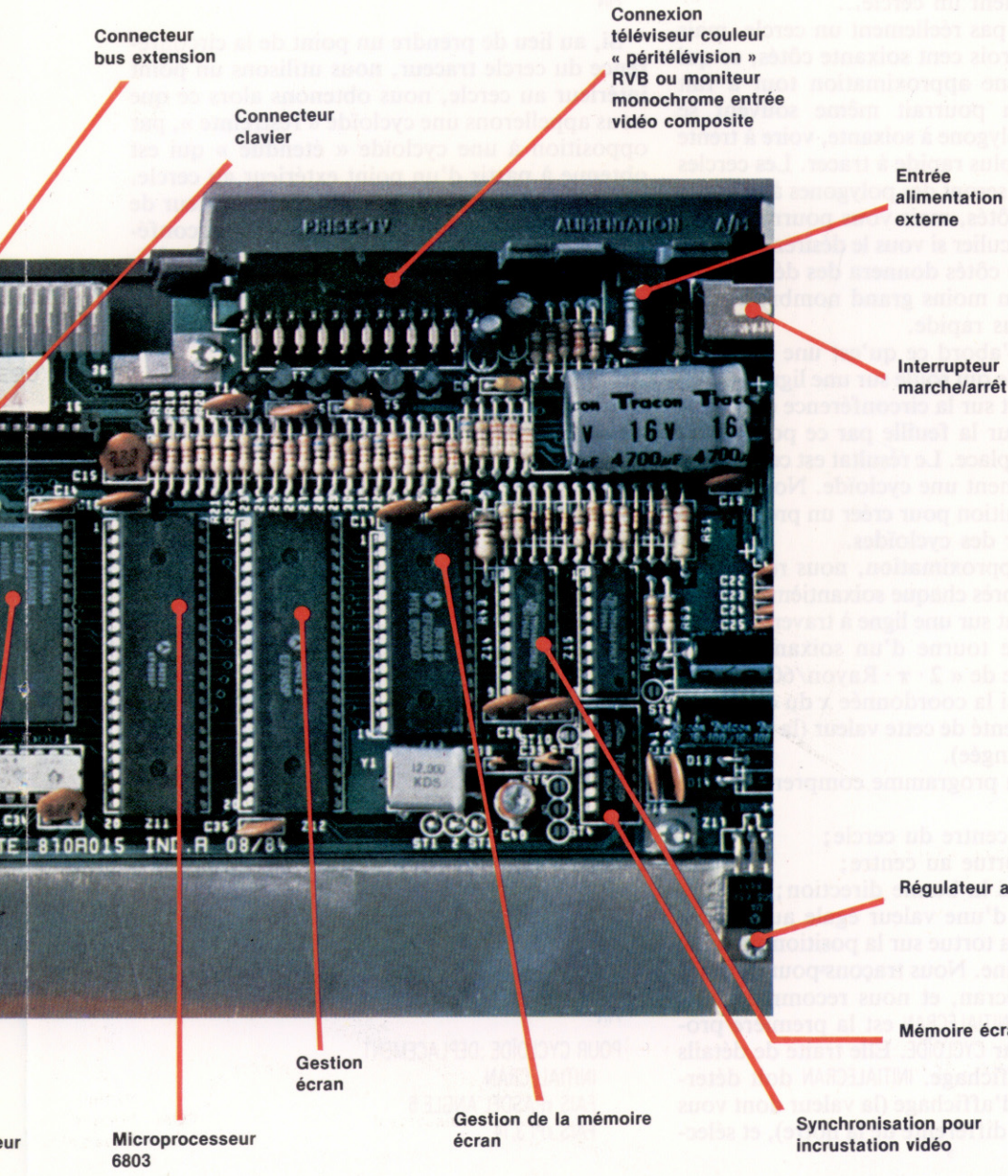


Le lecteur-enregistreur fonctionne sur piles ou alimentation 6 volts à l'aide d'un transformateur, et il s'accompagne d'un câble de raccordement.

Un ordinateur pour jouer et apprendre

Ce périphérique indispensable vous permettra non seulement de sauvegarder vos programmes, mais aussi et surtout de profiter de la large gamme de logiciels créés pour les Alice. Aux programmes déjà développés pour Alice 4 K — Jeu de dames, Annexion, Exercices de calcul, etc. — s'ajoute toute une variété de jeux d'aventures — « Casse-tête dans le métro », « Le sphinx d'or »... — et de didacticiels : les « Points BAC » en mathématiques et en français ; les jeunes musiciens pourront apprendre le solfège grâce à un petit poisson. D'autres jeux éducatifs s'adressent, selon le cas, aux très jeunes enfants ou aux lycéens. Au total, une trentaine de programmes ont été spécialement conçus pour Alice, et surtout pour le modèle 90, par les équipes de développement d'Édiciel, de Loriciels, d'Infogrammes et de Vifi-Nathan.

Commercialisé depuis la fin de l'année 1984, Alice 90 est vendu dans les magasins spécialisés, les grands magasins, les grandes surfaces. Il peut être acheté seul, au prix de 2 495 F, mais il est également disponible en tant que système complet vendu en coffret pour 1 000 F de plus. Une petite malette contenant l'Alice 90 et ses périphériques permet de les transporter aisément partout, à son domicile, à sa résidence secondaire, chez des amis. Le coffret comprend, outre l'Alice 90, les deux guides d'Alice, « Découvrez le BASIC » et « Les instructions de l'éditeur-assembleur », un lecteur-enregistreur, cinq cassettes de logiciels et tous les câbles de connexion.



Alice 90

PRIX

2 495 F TTC.
3 495 F pour le système complet en coffret.

UC

6803.

MÉMOIRE

40 K RAM, dont 32 K de mémoire vive utilisateur et 8 K vidéo.
16 K ROM pour le BASIC Microsoft incluant éditeur et éditeur-assembleur.

AFFICHAGE

16 lignes de 32 colonnes, ou 25 lignes de 40 ou 80 colonnes, pour le texte.
Résolution graphique : 160 × 125 pixels, et 320 × 250 pixels sous assembleur.

INTERFACES

Interface vidéo (Péritel), cassette et imprimante.

LANGAGE DISPONIBLE

BASIC

CLAVIER

AZERTY, type machine à écrire, 4 touches pour le déplacement du curseur, mots clés BASIC.

DOCUMENTATION

Les deux guides d'Alice sont fournis avec le coffret « Découvrez le BASIC » et « Les instructions de l'éditeur-assembleur ».

Logo à un fil

Nous commençons une courte série d'articles sur l'utilisation de LOGO pour la création de motifs géométriques. Nous allons tracer des cycloïdes, qui sont des courbes obtenues à partir de cercles.

Nous avons déjà vu, dans ce cours, comment utiliser LOGO pour tracer un cercle :

```
POUR CERCLE
  RÉPÈTE 360 [AVANCE 1 DROITE 1]
FIN
```

Le tracé est très laborieux et lent, même si on peut l'accélérer un peu en cachant la tortue. Si cette procédure ne donne pas un cercle sur votre écran mais une ellipse, vous devrez réinitialiser le rapport d'affichage : par essais successifs vous obtiendrez sûrement un cercle...

CERCLE ne trace pas réellement un cercle, mais un polygone de trois cent soixante côtés, ce qui est néanmoins une approximation tout à fait satisfaisante. On pourrait même souvent se contenter d'un polygone à soixante, voire à trente côtés, beaucoup plus rapide à tracer. Les cercles du présent article seront des polygones à soixante ou à cent vingt côtés, mais vous pourrez modifier ce point particulier si vous le désirez. Un plus grand nombre de côtés donnera des détails plus fins, tandis qu'un moins grand nombre entraînera un tracé plus rapide.

Voyons tout d'abord ce qu'est une *cycloïde*. Imaginez un cercle qui roule sur une ligne droite. Marquez un point sur la circonférence et relevez le chemin suivi sur la feuille par ce point alors que le cercle se déplace. Le résultat est ce que l'on appelle généralement une cycloïde. Nous allons utiliser cette définition pour créer un programme capable de tracer des cycloïdes.

En première approximation, nous relèverons des instantanés après chaque soixantième de tour d'un cercle roulant sur une ligne à travers l'écran. Lorsque le cercle tourne d'un soixantième de tour, il se déplace de « $2 \cdot \pi \cdot$ Rayon/60 » unités sur la ligne. Aussi la coordonnée x du centre du cercle aura augmenté de cette valeur (la coordonnée y reste inchangée).

La stratégie du programme comprend quatre actions simples :

1. déplacer le centre du cercle;
2. mettre la tortue au centre;
3. pointer dans la bonne direction;
4. la déplacer d'une valeur égale au rayon.

Ce qui amène la tortue sur la position suivante du point qui dessine. Nous traçons pour ce point un pointillé à l'écran, et nous recommençons.

La procédure INITIALÉCRAN est la première procédure appelée par CYCLOÏDE. Elle traite de détails nécessaires à l'affichage. INITIALÉCRAN doit déterminer le rapport d'affichage (la valeur dont vous aurez besoin sera différente de la nôtre), et sélectionner le mode NONENROULEMENT qui arrête le programme lorsque la courbe déborde de l'écran.

```
POUR DÉPLACECENTRE
  FAIS «XCENT :XCENT + :PAS
FIN

POUR POINTILLÉ
  POSEPLUME
  AVANCE 1
  RECULE 1
  LÈVEPLUME
FIN
```

Si, au lieu de prendre un point de la circonférence du cercle traceur, nous utilisons un point intérieur au cercle, nous obtenons alors ce que nous appellerons une cycloïde « restreinte », par opposition à une cycloïde « étendue » qui est obtenue à partir d'un point extérieur au cercle. Modifions CYCLOÏDE en prenant comme valeur de saisie la distance du point traceur à la circonférence. Les valeurs positives donnent les cycloïdes « restreintes », les valeurs négatives, les cycloïdes « étendues ».

```
POUR CYCLOÏDE
  INITIALÉCRAN
  FAIS «PASDEL'ANGLE 6
  FAIS «PI 3,14
  FAIS «RAYON 15
  FAIS «CIRCONFÉRENCE 2* :PI* :RAYON
  FAIS «PAS :CIRCONFÉRENCE / (360 / :PASDEL'ANGLE)
  FAIS «XCENT (-150)
  CYC 0
FIN

POUR INITIALÉCRAN
  RAPPORT 0,93
  NONENROULEMENT
  DESSINE
  LÈVEPLUME
  CACHETORTUE
FIN

POUR CYC :ANG
  DÉPLACECENTRE
  DONNEXY :XCENT 0
  DONNEDIRECTION :ANGLE
  AVANCE :RAYON
  POINTILLÉ
  CYC :ANG + :PASDEL'ANGLE
FIN

POUR CYCLOÏDE :DÉPLACEMENT
  INITIALÉCRAN
  FAIS «PASDEL'ANGLE 6
  FAIS «PI 3,14
```

```
    FAIS «RAYON 15
    FAIS «CIRCONFÉRENCE 2* :PI* :RAYON
    FAIS «PAS :CIRCONFÉRENCE / (360 / :PASDEL'ANGLE)
    FAIS «XCENT (– 150)
    FAIS «DISTANCE :RAYON – :DÉPLACEMENT
    CYC 0
FIN

    POUR CYC :ANG
    DÉPLACECENTRE
    DONNEXY :XCENT 0
    DONNEDIRECTION :ANG
    AVANCE :DISTANCE
    POINTILLÉ
    CYC :ANG + :PASDEL'ANGLE
FIN
```

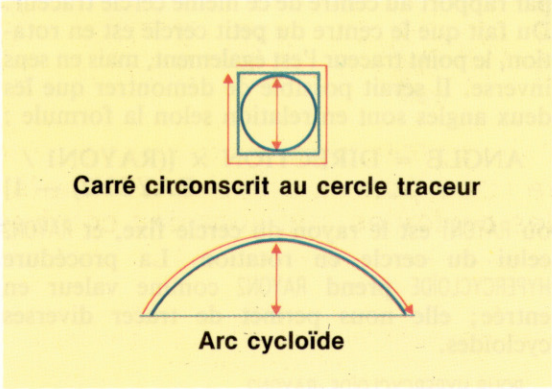
Réunir les points

Marquer les points successifs de pointillés permet une visualisation facile du tracé, mais les courbes seraient plus belles si nous pouvions réunir ces points en une ligne continue. La procédure RÉUNIS trace une ligne entre deux points :

```
POUR RÉUNIS :A :B
    DONNEPOS :A
    POSEPLUME
    DONNEPOS :B
    LÈVEPLUME
FIN

POUR DONNEPOS :POS
    DONNEXY PREMIER :POS DERNIER :POS
FIN
```

La procédure est utilisée avec les coordonnées des deux points mentionnés dans son appel. Un appel possible serait RÉUNIS [12 34][67 89]. Il nous faut garder trace dans notre programme de la dernière position du point, pour la relier à la position courante. L'état définitif de notre programme de tracé de cycloïdes devient :

```
POUR CYCLOÏDE :DÉPLACEMENT
    INITIALÉCRAN
    FAIS «PASDEL'ANGLE 6
    FAIS «PI 3,14
    FAIS «RAYON 15
    FAIS «CIRCONFÉRENCE 2* : PI* :RAYON
    FAIS «PAS :CIRCONFÉRENCE / (360/ :PASDEL'ANGLE)
    FAIS «XCENT (– 150)
    FAIS «DISTANCE :RAYON – :DÉPLACEMENT
    FAIS «ANCIENNEPOS LIST :XCENT :DISTANCE
    CYC 0
FIN

POUR CYC :ANG
    DÉPLACECENTRE
    DONNEXY :XCENT 0
    DONNEPOSITION :ANG
    AVANCE :DISTANCE
    FAIS «NOUVELLEPOS POS
    RÉUNIS :ANCIENNEPOS :NOUVELLEPOS
    FAIS «ANCIENNEPOS :NOUVELLEPOS
    CYC :ANG + :PASDEL'ANGLE
FIN

POUR POS
    RÉSULTAT LIST XCOR YCOR
FIN
```



Peut-être voulez-vous faire quelques expériences à partir de ces procédures. Par exemple, vérifiez le théorème qui dit que la longueur d'un arc d'une cycloïde est égale au périmètre du carré circonscrit au cercle traceur... Modifiez les procédures de tracé pour le vérifier.

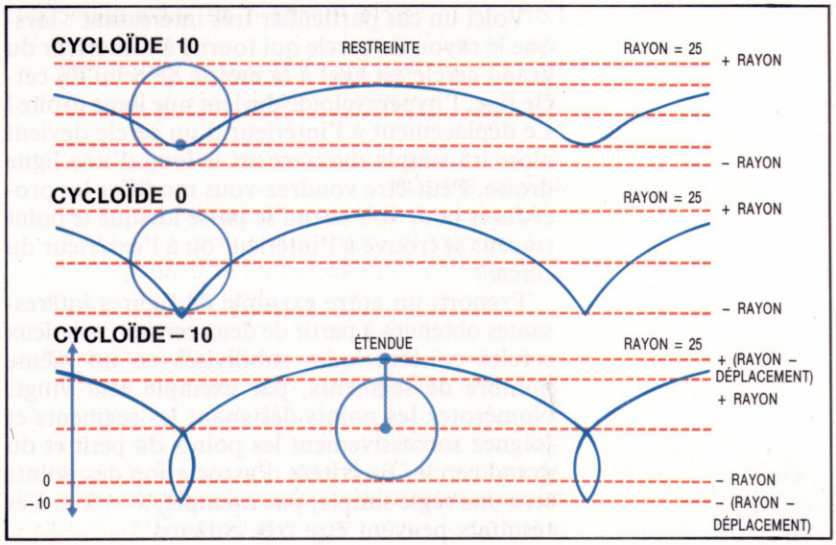
Si votre LOGO comporte des lutins, vous pouvez essayer une autre (et meilleure) méthode pour écrire le programme en définissant le point traceur comme lutin. L'avantage de cette méthode est de toujours savoir où se trouve le point, en utilisant DIS suivi de XCOR et YCOR.

Un petit plus

La création de cycloïdes (courbes tracées par un cercle roulant le long d'une ligne droite) n'est qu'une petite partie des ressources qu'offre un cercle en mouvement. Cependant, au lieu de se déplacer sur une ligne droite, le cercle traceur pourrait tourner à l'intérieur d'un autre cercle; le tracé du point situé sur le cercle interne s'appelle alors *hypercycloïde*.

Le problème reste le même dans ses grandes lignes : nous devons toujours déplacer le centre du cercle traceur, puis nous placer sur le point traceur de la circonférence. Il nous faut aussi tenir compte de deux angles. Le premier, DIRECTION, sert à déterminer la direction que prend le centre du cercle traceur; le second, ANGLE, donne l'angle dessiné par la direction du point traceur

Et roule...
Une cycloïde est obtenue par le déplacement d'un point traceur lié à un cercle en mouvement sur une ligne droite. La nature de la courbe varie selon que le point est situé à l'intérieur, à l'extérieur ou sur le périmètre du cercle. (Cl. Liz Dixon.)



par rapport au centre de ce même cercle traceur. Du fait que le centre du petit cercle est en rotation, le point traceur l'est également, mais en sens inverse. Il serait possible de démontrer que les deux angles sont en relation selon la formule :

ANGLE = DIRECTION × [(RAYON1 / RAYON2) — 1]

où RAYON1 est le rayon du cercle fixe, et RAYON2 celui du cercle en rotation. La procédure HYPERCYCLOÏDE prend RAYON2 comme valeur en entrée; elle nous permet de tracer diverses cycloïdes.

```
POUR HYPERCYCLOÏDE :RAYON2
   INITIALÉCRAN
   FAIS «PI 3,14
   FAIS «RAYON1 60
   FAIS «DIFFÉRENCE :RAYON1 – :RAYON2
   FAIS «DIRECTION 6
   FAIS «CIRCONFÉRENCE2 *:PI* :DIFFÉRENCE
   FAIS «VALEUR :CIRCONFÉRENCE / (360 / :DIRECTION)
   FAIS «ANGLE :DIRECTION * (:RAYON1 / :RAYON2 – 1)
   FAIS «CENTRE LISTE 0 :DIFFÉRENCE
   FAIS «DIRECTION 0
   FAIS «X POURCENT 0
   FAIS «ANCIENNEPOS LIST :XPOURCENT :RAYON1
   DIRECTIONCERCLE 0
FIN
POUR DIRECTIONCERCLE :ANG
   DÉPLACECENTRE2
   DONNEPOS POS
   DONNEDIRECTION :ANG
   AVANCE :RAYON2
   FAIS «NOUVELLEPOS POS
   RÉUNIS :ANCIENNEPOS :NOUVELLEPOS
   FAIS «ANCIENNEPOS :NOUVELLEPOS
   DIRECTIONCERCLE :ANG – :VALEURANGLE
FIN
POUR DÉPLACECENTRE2
   DONNEXY 0 0
   DONNEDIR :DIRECTION
   AVANCE :DIFFÉRENCE
   FAIS «CENTRE POS
   FAIS «DIR :DIR + :DIRECTION
FIN
```

Voici un cas particulier très intéressant : lorsque le rayon du cercle qui tourne à l'intérieur du grand cercle est égal à la moitié de celui du cercle fixe, l'hypercycloïde devient une ligne droite ! Le déplacement à l'intérieur d'un cercle devient alors un simple mouvement le long d'une ligne droite. Peut-être voudrez-vous modifier les procédures pour voir ce qui se passe lorsque le point traceur se trouve à l'intérieur ou à l'extérieur du cercle ?

Prenons un autre exemple de figures intéressantes obtenues à partir de deux cercles : soit deux cercles concentriques subdivisés en un même nombre de segments, par exemple cent vingt. Numérotez les points désignant les segments et joignez successivement les points du petit et du grand cercle. Le critère d'association des points sera une règle simple, par exemple, $x - 2x$. Les résultats peuvent être très curieux.

Cela peut se réaliser de manière concrète en cousant, mais aussi en dessinant, les deux cercles ; cela dit, nous préférerions que vous utilisiez tout simplement LOGO... Voici notre version d'un programme reliant des cercles :

```
POUR MISENPLACE
   FAIS «RAYONA 80
   FAIS «RAYONB 60
   DESSINE
   DIRECTION
   LÈVEPLUME
   RELIE 0 0
FIN
POUR RELIE :A :B
   SI :A = 120 ALORS STOP
   JOINS POINTA :A POINTB :B
   FAIS «A :A + 1
   FAIS «B 2 * A
   RELIE :A :B
FIN
POUR POINTA :NO
   DONNEXY 0 0
   DONNEDIR :NO * 3
   AVANCE :RAYONA
   RÉSULTAT POS
FIN
POUR POINTB :NO
   DONNEXY 00
   DONNEDIR NO * 3
   AVANCE :RAYONB
   RÉSULTAT POS
FIN
```

Avec d'autres règles de liaison entre les points ($x - 3x$, $x - 4x$, etc.), voyez à quoi ressemblent les nouvelles figures.

Variantes de logo

Pour toutes les versions LCSI :
La syntaxe SI est différente, par exemple :
SI :A = 120 [STOP].

DONNEPOS et POS existent en tant que primitives.

Remplacez DONNEXY par DONNEPOS (qui suppose une liste en entrée).

Utilisez CS pour TRACE.

Utilisez BARRIÈRE pour NONENROULEMENT (pour LOGO Atari, BARRIÈRE n'existe pas, remplacez-le par FENÊTRE et utilisez < BREAK > pour arrêter la procédure).

Pour donner le rapport d'affichage, utilisez :

.DONNEECR pour Atari;
DONNECRUNCH pour Apple;
DONNECRUNCH, suivi d'une liste, pour le Spectrum.

Étalonnage

Après les moteurs et les capteurs, nous allons étalonner le robot pour nous permettre de commander de façon précise les distances et les angles de ses déplacements.

Les moteurs pas-à-pas sont parfaitement adaptés aux dispositifs numériques : ils effectuent un pas de rotation précis chaque fois qu'ils reçoivent une impulsion. Afin d'ajuster le contrôle du moteur pas-à-pas avec le monde réel, nous devons effectuer quelques expériences servant à déterminer le nombre d'impulsions nécessaires pour déplacer le robot suivant les distances et les angles. Après ces expériences, nous devrions être en mesure de déterminer des rapports moyens impulsion/distance et impulsion/angle que nous pourrons entrer comme constantes dans des programmes. Ultérieurement, nous concevrons des programmes qui, entre autres, permettront au robot de se représenter des objets réels et d'en construire des représentations numériques.

Étalonnage linéaire

Nous pouvons faire une approximation du rapport impulsion/distance de notre robot au moyen de calculs élémentaires. Puisqu'une impulsion entraîne une rotation de 7,5° dans les moteurs, un rapport d'engrenage de 25/2 signifie qu'une impulsion provoquera une rotation de 0,6° (7,5 × 2/25) autour de l'axe. Puisque la roue Lego a un rayon de 30 mm, le mouvement linéaire par impulsion peut être calculé de la façon suivante : une impulsion provoque un mouvement de (0,6/360) × 2 × π × 30 mm. En simplifiant cette expression, une impulsion donne un mouvement de 0,1 × π mm. Nous obtenons ainsi une relation impulsion/distance : i/d = 3,183.

A chaque essai, le nombre d'impulsions et les distances théoriques sont affichés à l'écran. Vous pouvez vérifier la distance parcourue lors de chaque essai avec deux règles de 30 cm. Le programme affiche alors un tableau donnant le nombre d'impulsions, les distances réellement parcourues et les prévisions théoriques. Un rapport moyen i/d est également calculé. Cette valeur est importante. L'exemple de sortie de ce programme montre que notre robot se déplace un peu plus que ne laissaient prévoir les estimations.

```
 10 REM **** ETALONNAGE BBC ****
 20 DDR=&FE62:DATREG=&FE60
 30 ?DDR=15:REM SORTIE LIGNES 0-3
 50 forwards=4:backwards=2:DIM MD(12)
 60 FOR CC=500 TO 1700 STEP 100
 70 ?DATREG=0
 80 ?DATREG=(?DATREG OR 1) OR forwards
 90 PRINT CC,INT(CC*PI)/10
100 A$=GET$
110 FOR I=1 TO CC
120 PROCpulse
130 NEXT I
140 INPUT "DISTANCE EN MM" :MD((CC-500)/100)
150 NEXT CC
160 ?DATREG=0:T=0
180 PRINT"   IMPULSIONS","   MESUREES","   THEORIQUE","
200 FOR CC=500 TO 1700 STEP 100
210 PRINT CC,MD((CC-500)/100),INT(CC*PI)/10
220 T=T+CC/MD((CC-500)/100)
230 NEXT CC
240 PRINT:PRINT "RAPPORT IMPULSION SUR DISTANCE :",T/12
260 END
270 DEF PROCpulse
280 ?DATREG=(?DATREG OR 8)
290 ?DATREG=(?DATREG AND 247)
300 ENDPROC
```

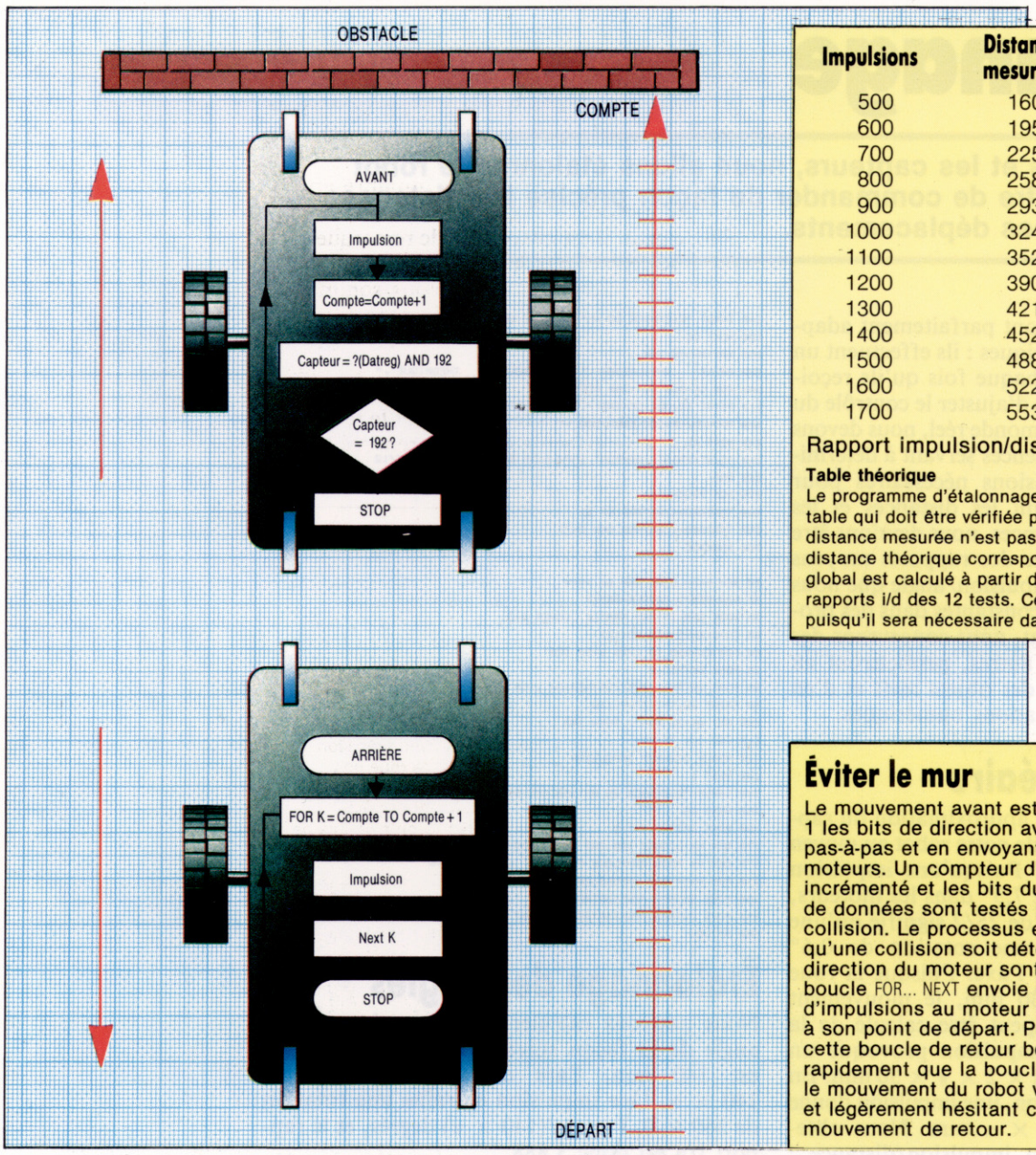
```
 10 REM **** ETAL CBM 64 ****
 20 DDR=56579:DATREG=56577
 30 POKE DDR,15:REM SORTIE LIGNES 0-3
 50 FW=4:BW=2:DIM MD(12)
 60 FOR CC=500 TO 1700 STEP 100
 70 POKE DATREG,0
 80 POKE DATREG,(PEEK(DATREG)OR 1)OR FW
 90 PRINT CC,INT(CC**)/10
100 GET A$:IF A$="" THEN 100
110 FOR I=1 TO CC
120 GOSUB 270:REM IMPULSION
130 NEXT I
140 INPUT "DISTANCE EN MM":MD((CC-500)/100)
150 NEXT CC
160 POKE DATREG,0:T=0
170 REM ** LIGNES 180-260 DE LA VERSION BBC **
175 REM ** MAIS REMPLACE PI PAR π LIGNE 210 **
270 REM **** S/P IMPULSION ****
280 POKE DATREG,PEEK(DATREG)OR 8
290 POKE DATREG,PEEK(DATREG)AND 247
300 RETURN
```

Étalonnage des angles

Nous pouvons calculer le rapport impulsion/angle de la façon suivante : si l'empattement est 140 mm, l'arc de cercle est égal à 140 × π. Une impulsion entraîne une rotation de 360 × 0, 1 × π /(140 × π) degrés, et le rapport i/a est donc 3,888.

L'un des principaux problèmes posés se situe au niveau de la précision des mesures. Puisque dans la plupart des applications le robot effectuera des rotations de 90° (ou des multiples de cette valeur), notre rapport théorique i/a nous indique que 389 impulsions seront nécessaires pour solliciter un virage de 90°.

Tracez deux lignes perpendiculaires sur une feuille de papier. Sur l'une des lignes, faites deux petites croix d'un côté ou de l'autre du point où les lignes se rencontrent pour désigner les points de départ des roues du robot. Exécutez le programme qui permet de faire effectuer une rotation de 90° au robot. La boucle FOR... NEXT de la ligne 70 définit le nombre d'impulsions transmises aux moteurs. Le nombre 371 est la valeur expérimentale permettant à notre robot d'effectuer une rotation de 90°. Éditez le programme, en modifiant la limite supérieure de cette boucle, jusqu'à ce que les roues de votre robot soient parfaitement alignées avec l'autre ligne perpen-



Kevin Jones

Impulsions	Distances mesurées	Distances théoriques
500	160	157
600	195	188,5
700	225	219,9
800	258	251,3
900	293	282,7
1000	324	314,1
1100	352	345,5
1200	390	376,9
1300	421	408,4
1400	452	439,8
1500	488	471,2
1600	522	502,6
1700	553	534

Rapport impulsion/distance : 3,34767511

Table théorique
Le programme d'étalonnage du robot produit cette table qui doit être vérifiée pour assurer que chaque distance mesurée n'est pas trop différente de la distance théorique correspondante. Un rapport i/d global est calculé à partir de la moyenne des rapports i/d des 12 tests. Ce nombre doit être noté puisqu'il sera nécessaire dans de futurs programmes.

Éviter le mur

Le mouvement avant est sollicité en mettant à 1 les bits de direction avant des moteurs pas-à-pas et en envoyant une impulsion aux moteurs. Un compteur d'impulsions est incrémenté et les bits du capteur du registre de données sont testés pour détecter une collision. Le processus est répété jusqu'à ce qu'une collision soit détectée, les bits de direction du moteur sont alors inversés et une boucle FOR... NEXT envoie un nombre suffisant d'impulsions au moteur pour renvoyer le robot à son point de départ. Puisque BASIC exécute cette boucle de retour beaucoup plus rapidement que la boucle impulsion-avant-test, le mouvement du robot vers l'avant est lent et légèrement hésitant comparé au mouvement de retour.

diculaire tracée sur le papier. D'autres vérifications peuvent être effectuées. Remplacez la direction « droite » (RT) par « gauche » (LF) à la ligne 60 et veillez aussi à ce que le robot effectue une rotation de 90° dans le sens inverse des aiguilles d'une montre. Doubler la valeur supérieure de la boucle FOR... NEXT provoquera une rotation de 180°. Veillez à ce que les roues reviennent au point de départ. Si ce n'est pas le cas, il est nécessaire d'effectuer un léger ajustement des roues pour qu'elles soient situées à des points symétriques par rapport à l'axe central. Lorsque la position des roues vous satisfait, marquez cette position sur l'axe et collez les roues en place.

```
10 REM **** CBM 64 ****
20 DDR=56579:DATREG=56577
30 POKE DDR,15:REM SORTIE LIGNES 0-3
40 IF=6:RT=0
50 POKE DATREG,0
60 POKE DATREG,(PEEK(DATREG)OR 1) OR RT
70 FOR I=1 TO 371:GOSUB 90:NEXT I
80 POKE DATREG,0:END
90 REM **** S/P IMPULSION ****
100 POKE DATREG,PEEK(DATREG)OR 8
110 POKE DATREG,PEEK(DATREG)AND 247
120 RETURN
```

```
10 REM **** BBC ****
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=15:REM SORTIE LIGNES 0-3
40 left=6:right=0
50 ?DATREG=0
60 ?DATREG=(?DATREG OR 1)OR right
70 FOR I=1 TO 371:PROCpulse:NEXT
80 ?DATREG=0:END
90 DEF PROCpulse
100 ?DATREG=(?DATREG OR 8)
110 ?DATREG=(?DATREG AND 247)
120 ENDPROC
```

```
10 REM **** PARE CHOCS CBM ****
20 DDR=56579:DATREG:56577
30 POKE DDR,15:REM SORTIE LIGNES 0-3
40 FW=4:BW=2
50 POKE DATREG,(PEEK(DATREG) OR 1) OR FW
60 REM **** IMPULSION AVANT ****
65 CC=0
70 GOSUB 1000:CC=CC+1:REM IMPULSION
80 IF (PEEK(DATREG) AND 192)=192 THEN 70
90 REM **** RETOUR AU DEPART ****
95 POKE DATREG,(PEEK(DATREG)AND 1)OR BW
100 FOR I=1 TO CC
110 GOSUB 1000:REM IMPULSION
120 NEXT I
```

```
130 POKE DATREG,0:END
1000 REM **** S/P IMPULSION ****
1010 POKE DATREG,(PEEK(DATREG)OR 8)
1020 POKE DATREG,(PEEK(DATREG)AND 247)
1030 RETURN

 10 REM **** PARE CHOCS BBC ****
 20 DDR=&FE62:DATREG=&FE60
 30 ?DDR=15:REM SORTIE LIGNES 0-3
 40 forwards=4:backwards=2
 50 ?DATREG=(?DATREG OR 1) OR forwards
 60 REM **** IMPULSION AVANT ****
 65 count=0
 70 REPEAT:PROCpulse:count=count+1
 80 UNTIL(?DATREG AND 192)  192
 90 REM **** RETOUR AU DEPART ****
 95 ?DATREG=(?DATREG AND 1) OR backwards
100 FOR I=1 TO count
110 PROCpulse
120 NEXT I
130 ?DATREG=0:END
1000 DEF PROCpulse
1010 ?DATREG=(?DATREG OR 8)
1020 ?DATREG=(?DATREG AND 247)
1030 ENDPROC
```

Maintenant que nous avons ajouté des capteurs à micro-interrupteurs à notre robot, nous pouvons écrire un logiciel qui utilise la sortie provenant du port utilisateur pour commander le robot et une entrée pour contrôler, à l'aide des capteurs, ses activités. Le simple programme suivant dirige le robot vers l'avant jusqu'à ce qu'il heurte un obstacle et le fait regagner son point de départ. La logique de ce programme peut être décrite de la manière suivante :

1. Mettre le registre de direction à 15. Cela met les bits 0-3 en sortie et les bits 4-7 en entrée.
2. Régler la direction du moteur vers l'avant.
3. Envoyer des impulsions aux moteurs jusqu'à ce que le bit 6 ou 7 passe au niveau bas, tout en comptant le nombre d'impulsions envoyées.
4. Mettre le moteur en marche arrière.
5. Envoyer un nombre déterminé d'impulsions.
6. Mettre à zéro le registre de données.

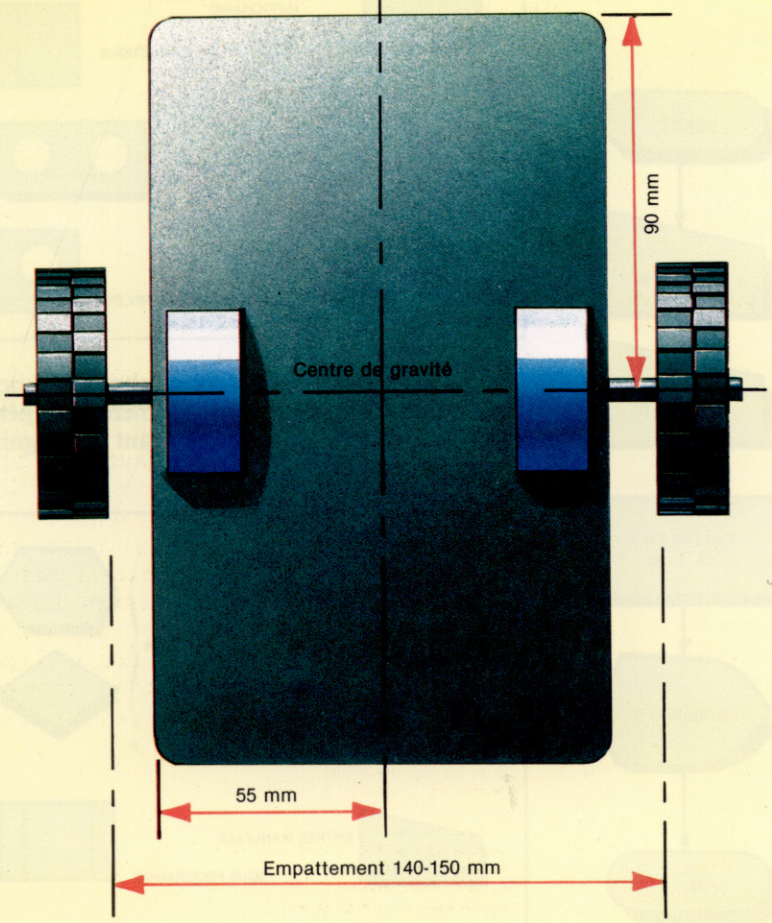
Dans ce programme, la paire de micro-interrupteurs avant est la paire située le plus loin des prises de connexion sur le couvercle du robot, et nous avons connecté ces deux micro-interrupteurs aux bits 6 et 7, à l'aide de deux cordons de connexion entre les deux paires de prises rouges et bleues situées à l'extrême droite du couvercle. Nous supposerons toujours que la fiche D est située plus à l'avant que les prises de connexion.

Parmi les quatre bits inférieurs du registre de données qui commandent le fonctionnement du moteur, le bit 0 est le bit de pause (normalement mis à 1), les bits 2 et 3 sont les contrôleurs de direction pour les moteurs de droite et de gauche, et le bit 4 envoie simultanément des impulsions aux deux moteurs, ce qui les fait avancer d'un pas lorsque celui-ci passe du niveau bas au niveau élevé. L'utilisation des opérateurs logiques AND et OR permet de mettre à 0 et à 1 les bits individuels, sans affecter les autres bits du registre. Comme les quatre bits supérieurs ont été retenus comme entrées par le registre de direction de données, ils sont normalement maintenus au niveau élevé. Lors de la fermeture d'un micro-interrupteur, le bit correspondant du registre de données passe au niveau bas. Normalement, les bits 6 et 7 ont la valeur 192 (= 128 + 64) s'ils sont mis en entrée. La boucle de répétition qui fait avancer le robot aux lignes 70-80 se termine lorsque ces deux bits n'ont plus la 192. Cela peut survenir si l'un des micro-interrupteurs est fermé (ou si les deux le sont). Si on fait un compte précis du nombre d'impulsions envoyées aux moteurs durant la période d'intervention, le robot peut alors regagner son point de départ en modifiant les bits de direction des moteurs et en envoyant un nombre approprié d'impulsions.

Finalement, il est intéressant de remarquer que le robot se déplace plus lentement vers l'avant que lorsqu'il revient sur ses pas. Nous sommes ici limités par la vitesse du BASIC. L'intervalle compris entre les impulsions de la boucle qui fait avancer le robot est plus long que celui des impulsions de retour, puisqu'un travail supplémentaire, comme le compte des impulsions et le test d'impact, doit être effectué dans la première boucle et non dans la deuxième.

Ajustement de l'empattement

Avant de commencer le processus d'étalonnage, il est nécessaire d'effectuer certains ajustements préliminaires. Mettez le robot sur le dos et localisez son axe central de rotation. Marquez l'axe central avec une petite encoche ou avec un trait de crayon indélébile. Mesurez la distance comprise entre l'intérieur des deux roues d'entraînement. Cette distance doit se situer entre 140 et 150 mm. Il est important, pour que le robot pivote avec précision autour de son axe central, que chaque roue soit équidistante du point d'axe central que nous venons de marquer. Les roues peuvent être déplacées délicatement le long de leurs axes jusqu'à ce que ce résultat soit obtenu.



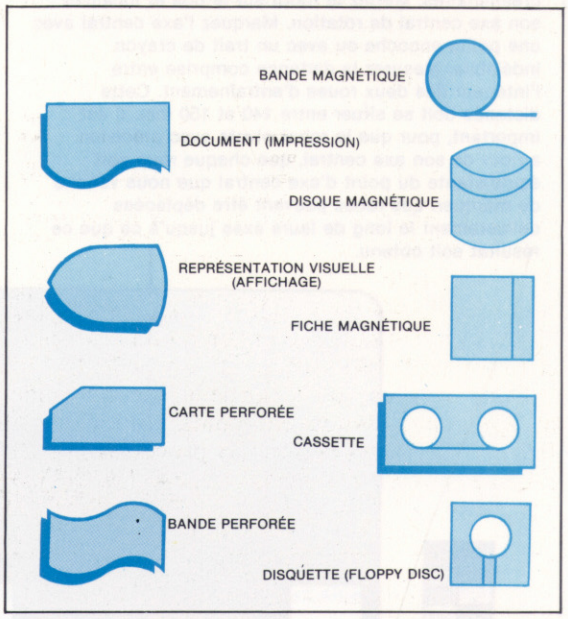
Kevin Jones

Symbolique

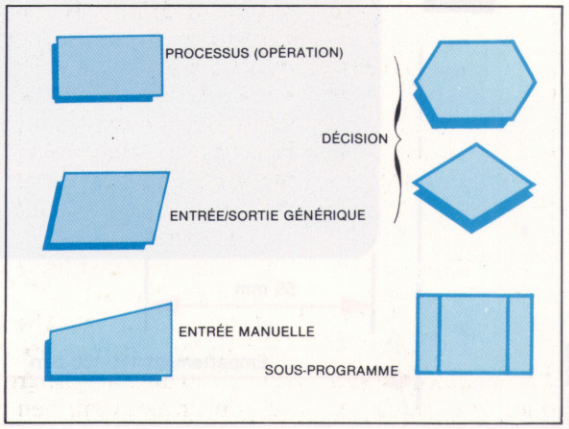
Nous réunissons ici des dessins géométriques qui nous servirons ultérieurement de symboles dans les organigrammes de flux, à mesure que ceux-ci deviendront plus complexes.

Les organigrammes se composent, comme nous l'avons déjà vu, d'une succession de symboles graphiques. Ceux-ci représentent, chacun de façon propre, les opérations que réalise l'ordinateur. Ces symboles peuvent se classer en quatre grands groupes : de système, de processus, auxiliaires et de lignes de flux.

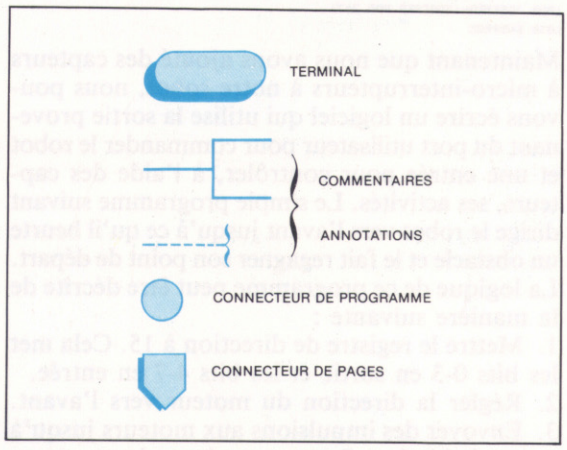
Symboles de système. Ils représentent physiquement les supports de données, qu'ils soient manuels ou automatiques. Ils traduisent également les unités et les composants des systèmes.



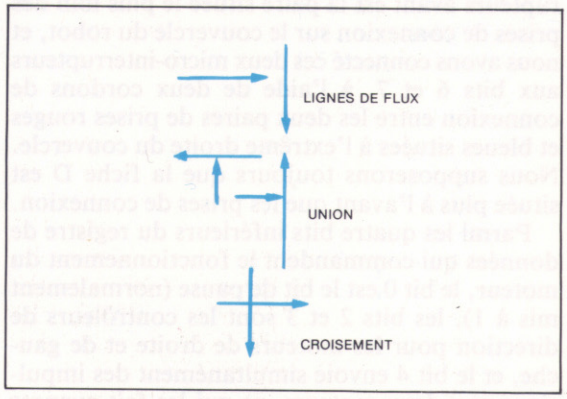
Symboles de processus. Ces symboles donnent une représentation du développement des opérations individuelles, tout en gardant une signification propre à l'ensemble.



Symboles auxiliaires. Ils sont utilisés pour donner une meilleure compréhension et plus de clarté à l'ensemble de l'organigramme.

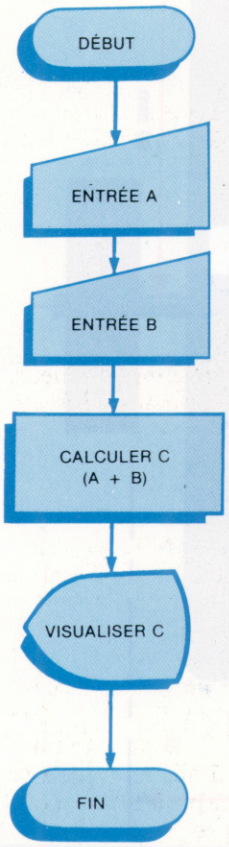


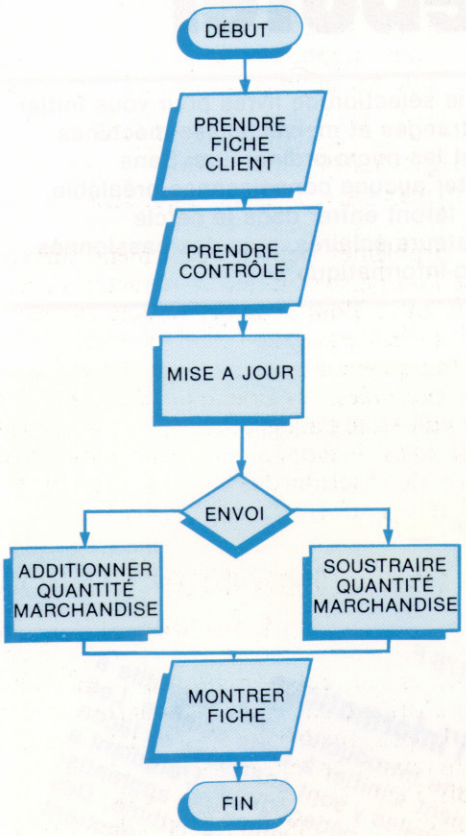
Lignes de flux. On pourrait les considérer comme un sous-ensemble du précédent ; elles marquent l'ordre des opérations et renvoient vers d'autres points de l'organigramme en les connectant entre eux.

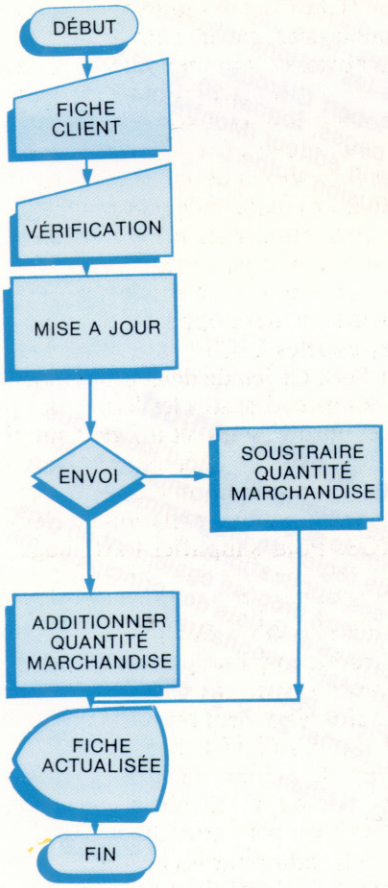


Peu à peu, et au fur et à mesure que les problèmes à résoudre apparaîtront, nous expliquerons ces symboles et autres combinaisons, ainsi que la fonction qu'ils accomplissent à l'intérieur de l'organigramme.

L'exemple de droite montre comment, après l'entrée de deux valeurs par le clavier, on arrive à visualiser la valeur de la somme de ces quantités. Il sert également à montrer le symbole appartenant au groupe d'auxiliaires connus sous le nom de « terminal ». Ce symbole s'applique indistinctement au départ (START) et à la fin (END). L'utilisation de ce symbole est obligatoire dans tous les organigrammes.







Symboles d'entrée et de sortie

Pour comprendre l'utilisation de ces symboles, partons d'un exemple simple de comptabilité : « Les clients d'une entreprise ont leurs fiches modifiées (fiches où sont enregistrés des renseignements divers comme les raisons sociales, les mouvements de marchandises, les paiements, etc.), conformément aux contrôles effectués quotidiennement par les services compétents de l'entreprise. C'est là une tâche tout à fait classique qui, pendant longtemps, s'est passée des services de l'ordinateur. Cela dit, il est difficile aujourd'hui, dans l'économie moderne, d'ignorer cet outil si l'on veut rester compétitif.

» Dans ces fiches apparaît la quantité de marchandise en mouvement qui, s'il s'agit d'un envoi, devra s'additionner à la marchandise existante apparaissant dans la fiche, alors que, s'il s'agit d'un retour, la quantité devra être soustraite. Dans n'importe quel cas, on établira une fiche comme correspondant à un processus complet dûment actualisé. »

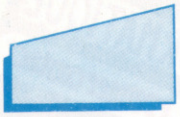
Après analyse du problème, on se rend compte qu'un même type d'opération apparaît plusieurs fois ; on peut remarquer en effet deux entrées différentes : celle de la fiche concernant le client et celle des contrôles. Étant donné que l'on ne s'informe pas expressément de la nature de ces entrées, celles-ci seront considérées comme génériques.

Ainsi, quand on lit « prendre état de contrôle », on comprendra « donner le chiffre qu'il contient », c'est-à-dire la quantité correspondante de marchandise dans le cas de la fiche du client.

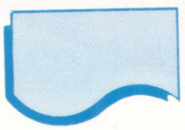
Dans la phase de « sortie », on demande qu'apparaisse la nouvelle configuration de la fiche, dûment actualisée. Le processus qui en découle permet de déterminer s'il s'agit d'un renvoi ou non.

E.S. générique
Représente une fonction d'entrée/sortie, c'est-à-dire un mouvement des données se rapportant soit à l'introduction d'une information en mémoire nécessaire à l'élaboration d'un processus, soit l'obtention de renseignements qui font partie d'une information déjà élaborée (voir fig. 1).

Affichage
Constitue la représentation visuelle sur l'écran des données et des résultats (voir fig. 2)



Entrée manuelle
Implique une entrée des informations par le clavier directement sous l'intervention de l'opérateur (voir fig. 2).

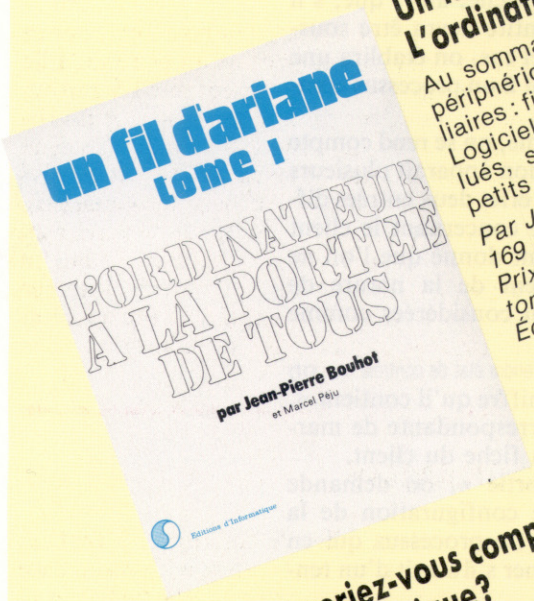


Document
Traduit une fonction d'entrée/sortie ayant un document comme support. On l'utilise en général pour signifier l'usage d'une imprimante.

Livres pour débuter

Si vous ne savez pas au juste ce qu'est l'informatique...
Si vous n'avez jamais touché à un ordinateur...
Si vous souhaitez acheter un micro...
Si vous l'avez mais ne savez pas vous en servir...

Voici une sélection de livres pour vous initier à ces étranges et merveilleuses machines que sont les micro-ordinateurs. Sans nécessiter aucune connaissance préalable, ils vous feront entrer dans le cercle des amateurs éclairés, puis des passionnés de micro-informatique.



Un fil d'Ariane, L'ordinateur à la portée de tous

Au sommaire : ordinateur et information. Les périphériques d'entrée/sortie. Mémoires auxiliaires : fichiers, etc. L'évolution des ordinateurs. Logiciel : composants de base, langages évolués, systèmes d'exploitation. Les nouveaux petits ordinateurs. Index de 1 000 mots.

Par J.-P. Bouhot et M. Peju.
169 pages, format 21,5 × 27,5 cm.
Prix : tome 1, 97 F ; tome 2, par J.-P. Bouhot, Cottin et Tricot, 176 F.
Édition d'informatique.

Initiation à l'informatique

Cette excellente introduction sera très utile à ceux qui veulent s'initier à cette science. Les notions essentielles y sont analysées de façon très détaillée et très pédagogique. L'étudiant a la possibilité d'évaluer lui-même son apprentissage et d'avancer à son propre rythme. Des exemples tirés de la vie courante illustrent toutes les notions introduites.

Par Robert Cléroux.
485 pages, format 20 × 22,5 cm.
Guérin éditeur (Montréal).
Diffusion Vuibert.

Aimeriez-vous comprendre l'informatique ?

Ce guide simple et clair, appuyé par de nombreuses illustrations, présente une série d'applications telles que l'utilisation de l'ordinateur pour les images de la météo, le contrôle du trafic aérien, les nouvelles machines à laver, le traitement de texte, ainsi qu'une initiation « en douceur » au langage de programmation BASIC.

Par Bradbeer, de Bono et Laurie.
220 pages, format 17 × 21,5 cm.
Prix : 98 F.
Inter-Éditions.

L'informatique, premier contact

Un auteur plein de fantaisie et plein d'idées vous fait découvrir avec le sourire l'informatique : son vocabulaire, ses règles de programmation, son utilisation et ses applications.
Ce fascicule illustré propose également un glossaire informatique et la liste des principales instructions du BASIC à connaître.

Par Jean-Pierre Petit.
84 pages, format 21 × 21 cm.
Prix : 42 F.
C.E.D.I.C.-Nathan.

Page manquante
(publicité)

Page manquante
(publicité)